UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Vedran Jašarević

# REKONSTRUKCIJA APLIKACIJE ZA VODENJE POSLOVANJA RESTAVRACIJE

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

Mentor: izr. prof. dr. Marjan Krisper

Ljubljana, 2016

Naslov: Rekonstrukcija aplikacije za vodenje poslovanja restavracije

Tematika:

V okviru diplomske naloge izdelajte arhitekturni model aplikacije za vodenje restavracije razvite z agilnim pristopom in metodo ekstremnega programiranja (Extreme Programming - XP). Uporabite metodo rekonstrukcije (Software Architecture Reconstruction Method).

Z uporabo UML diagramov, opisov tabel in dokumentacije na različnih ravneh abstrakcije omogočite lažje vzdrževanje, odpravo napak in zamenjavo razvijalcev in vzdrževalcev programske opreme. Končni cilj naj bo tudi odstranitev podvojene kode, optimizacija modulov, poenostavitev uporabe in povečanje prilagodljivosti aplikacije.

# IZJAVA O AVTORSTVU

## diplomskega dela

Spodaj podpisani **Vedran Jašarević**,

z vpisno številko **63020338**,

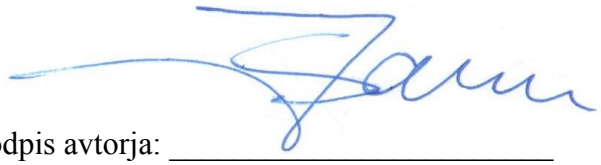
sem avtor diplomskega dela z naslovom:

**Rekonstrukcija aplikacije za vodenje poslovanja restavracije**

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom

  **izr. prof. dr. Marjana Krisperja**
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.



V Ljubljani, dne 12.04.2016 Podpis avtorja: _____

# Zahvala

Hvala mentorju izr. prof. dr. Marjanu Krisperju za strokovno pomoč pri izdelavi diplomskega dela, za nasvete in usmerjanje.

Rad bi se zahvalil tudi družini in prijateljem za vso podporo in spodbudo pri zaključevanju študija.

Hvala vsem ostalim, ki ste mi vsa leta stali ob strani.

# Kazalo vsebine

# Kazalo slik

# Kazalo tabel



# Seznam uporabljenih kratic in simbolov

Armin – Architecture Reconstruction and Mining

CLR – Common Language Runtime

DURS – Davčna uprava Republike Slovenije

EOR – enkratna identifikacijska oznaka računa

GreQL – Graph Repository Query Language

GXML – Graphical eXtensible Markup Language

IL – Intermediate Language

InfoS POS – aplikacija za vodenje poslovanja v gostinstvu

JIT – Just In Time Compilation

MDE – Model Driven Engineering

MSSQL – Microsoftova podatkovna baza

POS – Point Of Sale

RSF – Rigi Standard Format

SA – Software Architecture

SOAP – Simple Object Access Protocol

UML – Unified modeling language

XP – ekstremno programiranje

XML – EXtensible Markup Language

XSD – XML Schema Definition

## Povzetek

V diplomskem delu je opisana izdelava modela že razvite programske opreme za vodenje poslovanja restavracij, gostiln in podobnih objektov. Uporabljen je način rekonstrukcije aplikacije od spodaj navzgor (*bottom-up*), pri katerem se že iz delujočih in razvitih modulov izdelajo modeli z različnimi diagrami in tehnikami na različnih nivojih abstrakcije. Na začetku sta opisana problemska domena in ozadje izdelave aplikacije ter razlogi za rekonstrukcijo oz. izdelavo modela. V nadaljevanju sledi opis uporabljenih orodij, jezikov, metod in tehnik. Navedeni so različni modeli, uporabljeni pri modeliranju programske opreme (razredni diagrami, primeri uporabe, zaporedni diagrami, diagrami postavitve, opis podatkovne baze in tabel ...). Nazadnje so napisane možne izboljšave, ki so razvidne iz izdelanega modela in ki so bile spregledane zaradi agilnega pristopa pri razvoju, in metode ekstremnega programiranja aplikacije.



Ključne besede: model razvoja programske opreme, rekonstrukcija, aplikacijski sistem restavracije

## Abstract

The document describes the Software Architecture Reconstruction Method of already developed application for the management of operations in restaurants, bars and similar facilities. It uses a bottom - up reconstruction application where the modelling is based on analysis of existing modules in order to depict different diagrams and techniques at different levels of abstraction. At the beginning, the problem domain is described together with the background of development process, as well as the reasons and motives for the reconstruction i.e. model making. This is followed by the description of tools, methods and techniques. Next described are different models used in the modelling process (class diagrams, use cases, sequence diagrams, layout diagrams, description of the database and tables). Finally described are the possible improvements reflected in the developed model and which were overlooked due to the agile approach of development and extreme programming methods.



Key words: Design recovery, Reverse engineering, Software Architecture Modeling, Restaurant management application

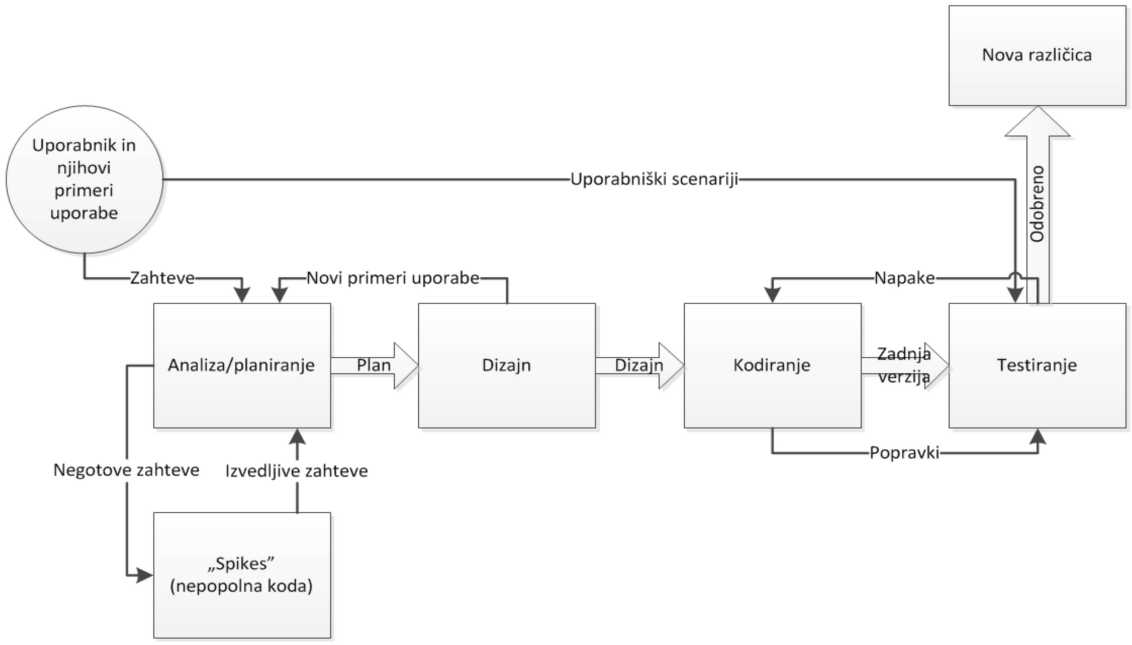
# 1 Uvod

## 1.1 Problemska domena

Leta 2008 je bila razvita aplikacija, imenovana InfoS POS (www.pos-blagajne.com) za vodenje poslovanja gostiln, restavracij, pekarn in podobnih poslovnih objektov. Motiv za izdelavo aplikacije je bil, da so bile obstoječe rešitve zastarele (npr. register blagajne ali sistemi DOS, pri katerih je bilo vzdrževanje časovno potratno), preveč kompleksne ali pa so bili njihovi razvijalci toliko togi, da niso dovolili nikakršnih dodatnih sprememb, ki so jih zahtevale stranke. Dogovor z nekaterimi kupci, ki so bili nezadovoljni z dotedanjimi rešitvami, je zahteval, da mora biti prva delujoča verzija izdelana že v nekaj tednih. Izdelava prototipa je zahtevala agilen pristop in izbrano metodologijo XP [10] ekstremnega programiranja. Aplikacija je v začetku vsebovala samo modul za izdajanje računov, nato je bil dodan modul za upravljanje zaloge, z leti pa se je zaradi dodatnih zahtev nadgrajevala z moduli za kosovnice, naročanje, oddaljen pristop, za naprave, občutljive na dotik, ter z raznimi pregledi in statističnimi podatki, trgovsko knjigo, sistemom *happy hour*, meniji, sinhronizacijo v oblak (*cloud*) itd. Zaradi nenehno prisotnih zahtev s smeri kupcev in stalno spreminjajoče se zakonodaje je bil življenjski cikel razvoja skrajšan na najkrajši možni čas, cikli pa so se ponavljali v mnogo iteracij. Nove različice aplikacije so se izdajale tedensko ali pa tudi dnevno. Zaradi izbrane metodologije so trpele določene komponente, ki so še kako potrebne pri razširitvi ali zamenjavi razvojne, podporne in tržne skupine.

Aplikacija je sedaj prišla do zrele faze, kjer so več ali manj zadovoljene potrebe vseh trenutnih uporabnikov. Nove napake se ne pojavljajo oz. se zelo redko. Nove različice se ne izdajajo več na dnevni ali tedenski ravni, ampak enkrat na dva meseca, ter je dovolj časa za izboljšave in dokumentacijo. Aplikacijo se trenutno uporablja v več kot 400 različnih objektih na področju Hrvaške in Slovenije.

Vse našteto je zadosten motiv za izdelavo modela omenjene aplikacije, ki bo v prihajajočih časih v prid vsakemu, ki bo sodeloval v razvoju, podpori in trženju. Model definira obravnavani sistem tako, da je lažje razumljiv, omogoča lažjo komunikacijo in je bližje ljudem, ki ga uporabljajo.




Slika 1: Diagram metode ekstremnega programiranja [10].

## 1.2 Cilji diplomskega dela

Pridobljeni model aplikacije bo kot končni produkt omogočal reševanje nekaj problemov in s tem doseganje več različnih ciljev.

Najpomembnejši cilji diplomskega dela so:

- omogočiti lažjo zamenjavo/vpeljavo članov razvojne skupine,
- olajšati partnerjem (podjetjem, ki prodajajo obravnavano aplikacijo) razumevanje delovanja in s tem posredno povečati prodajo,
- omogočiti lažjo podporo končnim kupcem in hitrejše reševanje problemov,
- optimizirati aplikacijo in izločiti podvojeno kodo,
- prepoznati nove funkcionalnosti, ki bodo povečale dodano vrednost aplikacije.

# 2 Opis pristopov, metod in tehnologij

V nadaljevanju so podani modeli, tehnologije, orodja in različni pristopi pri izdelavi modela iz že razvite aplikacije oz. postopek rekonstrukcije modela aplikacije [6].

Aplikacija je bila v celoti razvita z razvojnim orodjem Microsoft Visual Studio [7]. Prednost tega orodja je, da vsebuje orodja za povezovanje s podatkovnimi bazami, izdelavo namiznih aplikacij, spletnih aplikacij, spletnih servisov, jezika za modeliranje (UML [9]), gre za prijazen uporabniški vmesnik in ima velik nabor orodij, ki skrajšajo razvoj aplikacije, skratka, več ali manj vse, kar zadošča za razvoj aplikacije takšnega obsega. Kot programski jezik pri razvoju in vzdrževanju se uporablja C# z nekoliko povezanih zunanjih neupravljanih knjižnic.

Kot podatkovna baza je bila v začetku uporabljena baza MS Access, kasneje pa je bila zaradi potrebe po večji hitrosti in zanesljivosti ter delu v mrežnem okolju dodana podpora za MS SQL.

Pogoj za delovanje aplikacije je nameščeno ogrodje Microsoft .NET [12]. Ogrodje Microsoft .NET Framework in razvojni sistem Microsoft Visual Studio kot skladna celota omogočata hitro izdelavo uporabnikom prijaznih programov. Vgrajeno ima množico orodij za razvoj varnih, hitrih, dopadljivih aplikacij različnih tipov, ki bode ustrezale sodobnim zahtevami. S temi orodji omogoča varno povezovanje na spletne servise. Ogrodje .NET Framework 4.5 je nazaj kompatibilno in lahko na istem računalniku obstaja več različnih verzij ogrodja, ki delujejo paralelno tako, da je zagotovljena tudi podpora za starejše aplikacije, ki so razvite za starejše verzije ogrodja.

Kot jezik modeliranja je izbran UML (*Unified Modeling Language*) [9], ki je opisan v nadaljevanju.

## *2.1 UML*

UML je standardiziran vizualni jezik za modeliranje in je prvotno namenjen:

- modeliranju poslovanja in pripadajočih procesov,
- analizi, dizajnu in implementaciji programskih rešitev.

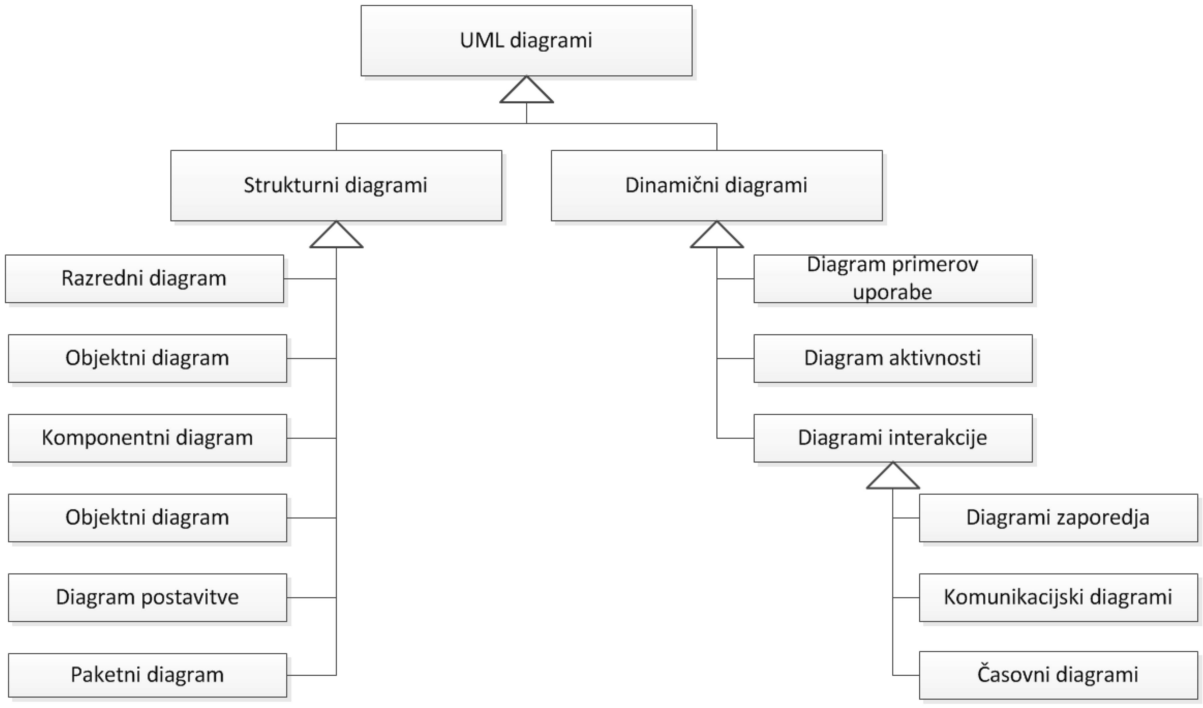
UML je standardiziran jezik za vizualno modeliranje, ni pa jezik za beleženje in razvoj procesa razvoja programske opreme. Je namenoma neodvisen od tipa procesov in se lahko uporablja za različne procese, čeprav je najbolj primeren pri uporabniško usmerjenih, iterativnih in inkrementalnih procesih.

UML ni v celoti grafični in vizualni jezik. S pregledovanjem določenega diagrama včasih ni možno razumeti delovanja in narave sistema samo s slike, zato se po navadi z vsakim diagramom poda tudi opis tistega dela sistema, na katerega se nanaša obravnavani diagram.

UML je splošno namenjen za izdelavo in prikaz modelov na področju razvoja programske opreme. Predstavlja *de facto* standard za vizualizacijo dizajna sistema ali aplikacije. Zgodovina sega v leto 1994, ko so ga razvili Grady Booch, Ivar Jacobson in James Rumbaugh v podjetju Rational Software. Leta 1997 je bil razvoj jezika UML prestavljen pod organizacijo Object Management Group (OMG) in se od takrat dalje tam tudi razvija. Leta 2005 ga je organizacija ISO sprejela kot standard za modeliranje arhitekture programske opreme. Nenehno se spreminja in dopolnjuje, vendar rdeča nit ostaja ista. Ponuja več različnih diagramskih tehnik za vizualni prikaz modela dela ali celotnega sistema, ki omogoča različne poglede na sistem in delovanje sistema.

Modeliranje z jezikom UML lahko razdelimo v dve glavni skupini:

- Statični pogled (struktura sistema) – prikazuje strukturo sistema z uporabo objektov, operacij, atributov, komponent, medsebojnih povezav in odvisnosti. Za statični pogled se uporabljajo razredni, konceptualni, objektni in komponentni diagrami ter diagrami postavitve. Elementi strukturnih diagramov predstavljajo smiselne koncepte sistema, ki lahko vključuje abstraktne koncepte, koncepte iz realnega sveta in koncepte implementacije. Strukturni diagrami se ne ukvarjajo s časovno naravnanimi koncepti.
- Dinamični pogled (obnašanje sistema) – prikazuje obnašanje sistema, sodelovanje med posameznimi moduli, spremembe stanja objektov in atributov. Uporabljajo se diagrami primerov uporabe, diagrami aktivnosti, diagrami zaporedja, časovni diagrami. Prikazujejo spremembe sistema skozi čas.




Slika 2: Hierarhični prikaz različnih diagramov UML [8].

Specifikacija UML dovoljuje tudi kombiniranje različnih diagramov za prikaz želenega koncepta, vendar običajno orodja za izdelavo diagramov UML tega ne dovoljujejo.

## 2.2 Microsoft Visual Studio

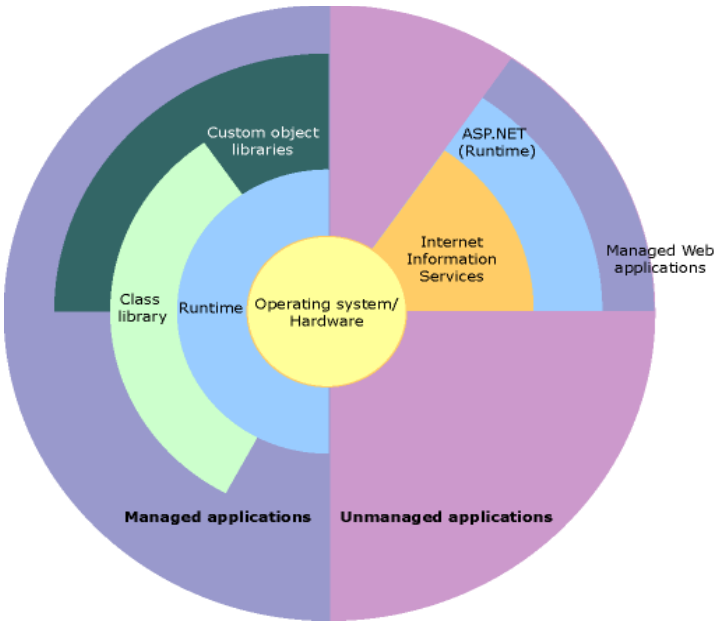
Celotna aplikacija je izdelana v razvojnem okolju Microsoft Visual Studio. Ker v zadnjih verzijah okolje vsebuje tudi orodja za modeliranje UML, je le-ta uporabljen tudi za izdelavo modelov.

Microsoft Visual Studio ponuja celovito razvojno okolje za izdelavo sodobnih aplikacij za okolja Windows, Android in iOS. Omogoča tudi razvoj sodobnih spletnih aplikacij in servisov v oblaku. Vsebuje orodja za dizajniranje, urejanje, modeliranje, razhroščevanje in podporo za različne programske jezike (C#, Visual Basic, F#, C + +, HTML, Javascript, Phyton in še več).

Omogoča izdelavo modela arhitekture programske opreme. Iz že napisane kode se lahko avtomatsko generirajo razredni diagrami. Gradnike razrednih diagramov je treba po generiranju še ročno povezati.

.Net Framework [12] je komponenta operacijskih sistemov Windows, ki omogoča upravljano virtualno izvajanje na sistemu CLR (*Common Language Runtime*) in podporo izvajanja programske opreme, napisane v različnih jezikih. Sestavljena je še iz knjižnice razredov uporabnih komponent (*Class Libraries*). CLR skrbi pri izvajanju upravljane kode za upravljanje s pomnilnikom, varno izvajanje kode, izvajanje niti, prevedbo in za ostale sistemske storitve. Class Libraries nudi uporabo velike množice že napisanih kod, da ne zapravljamo časa za pisanje nižje nivojskih pogosto rabljanih operacij.

Slika 3 prikazuje odnos komponent CLR in Class Library proti aplikacijam in sistemu v celoti. Prikazuje tudi, kako upravljana koda deluje znotraj večje arhitekture.

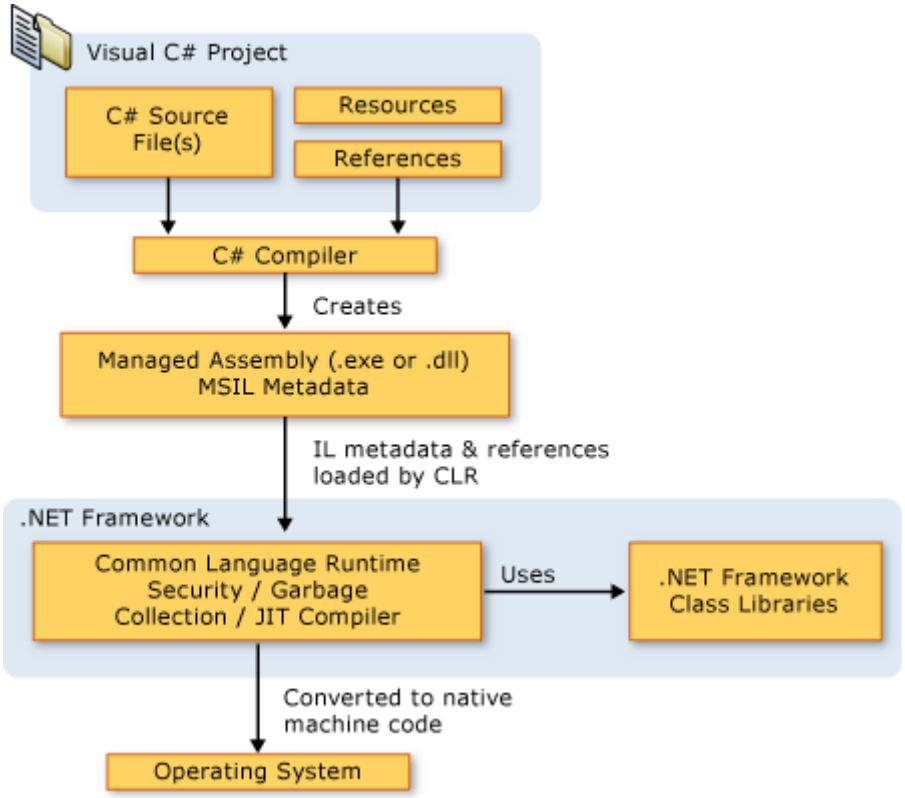



Slika 3: Kontekst ogrodja .Net Framework [12].

Izvorna koda, napisana v enem od jezikov, ki jih podpira Visual Studio, se prevede v vmesni jezik (IL – angl. *Intermediate Language)*. Koda IL in ostali viri se shranijo v obliki sklopa – datoteke (največkrat s končnico .exe ali .dll). Datoteka vsebuje manifest, v katerem so shranjeni podatki o verziji, skupini, varnostnih zahtevah in jeziku. Ko se program C# zažene, se sklop naloži v CLR, ki naredi različne akcije v odvisnosti od podatkov v manifestu. Če so vse zahteve izpolnjene, bo CLR naredil prevajanje v realnem času (JIT – angl. *Just In Time Compilation*) in prevedel kodo IL v strojno kodo.

V programskem jeziku C# seveda lahko pišemo tudi neupravljano kodo (podobno C + +), kjer moramo samostojno poskrbeti za pomnilnik in podobne zadeve, pomembne v nižjih jezikih.

Slika 4 prikazuje povezave med aplikacijo C# in komponentami .Net.




Slika 4: Arhitektura platforme .Net Framework [11].

## 2.3 Izbira pristopa

Obstaja več različnih obstoječih pristopov [5], ki se uporabljajo pri rekonstrukciji aplikacije.

Skupine pristopov in orodij lahko razdelimo na:

- ročno rekonstrukcijo,
- ročno rekonstrukcijo s pomočjo orodij,
- povpraševalni jezik za izdelavo vzorcev, iz katerih se avtomatsko izdela agregiran model,
- uporabo ostalih tehnologij (podatkovno rudarjenje, povezovanje, jeziki za opis arhitektur).

V diplomskem delu je bil izbran prvi pristop ročne rekonstrukcije, pa čeprav VS vsebuje orodje za rekonstrukcijo razrednih diagramov, so bili ti izdelani in povezani ročno.

Razlogov odločitve za prvi pristop je več, nekateri so našteti spodaj:

- dobro poznavanje kode prisotnih razvijalcev,
- dobro poznavanje podatkovne strukture prisotnih razvijalcev,
- dobro poznavanje problemske domene prisotnih razvijalcev,
- relativno majhen obseg kode,
- dobro poznavanje vseh možnosti aplikacije in možnih postavitev sistema v različnih okoljih ter pri različnih zahtevah,
- ročna rekonstrukcija je hkrati tudi najbolj preprost pristop,
- kompleksnost ostalih treh pristopov, čas, potreben za spoznavanje z orodji, nedoslednost orodij,
- omogoča najbolj detajlno in realno predstavitev sistema na različnih nivojih abstrakcije,
- aplikacija je v celoti razvita v programskem jeziku C#, zato je pregledovanje kode veliko lažje,
- razredni diagram je hkrati tudi entitetni diagram oz. model podatkovne baze.

## 2.4 Metodologija

Obstaja več različnih metodologij, ki jih lahko izberemo pri rekonstrukciji aplikacije, vendar so po navadi vse naravnane za točno določene namene in z uporabo točno določenih orodij. V ospredje so se kot najbolj splošne postavile naslednje:

- simfonija [1],
- kakofonija [2].

Tako simfonija kot kakofonija sta zelo generični in splošni metodologiji za rekonstrukcijo. V veliki meri se prekrivata in sta si podobni. Ne določata natančnih tehnologij in postopkov, ki si jih lahko izvajalci procesa rekonstrukcije poljubno izberejo glede na različne faktorje.

Simfonija se osredotoča na vidike in poglede, kakofonija pa na metamodele in povezavo z arhitekturo programske opreme.

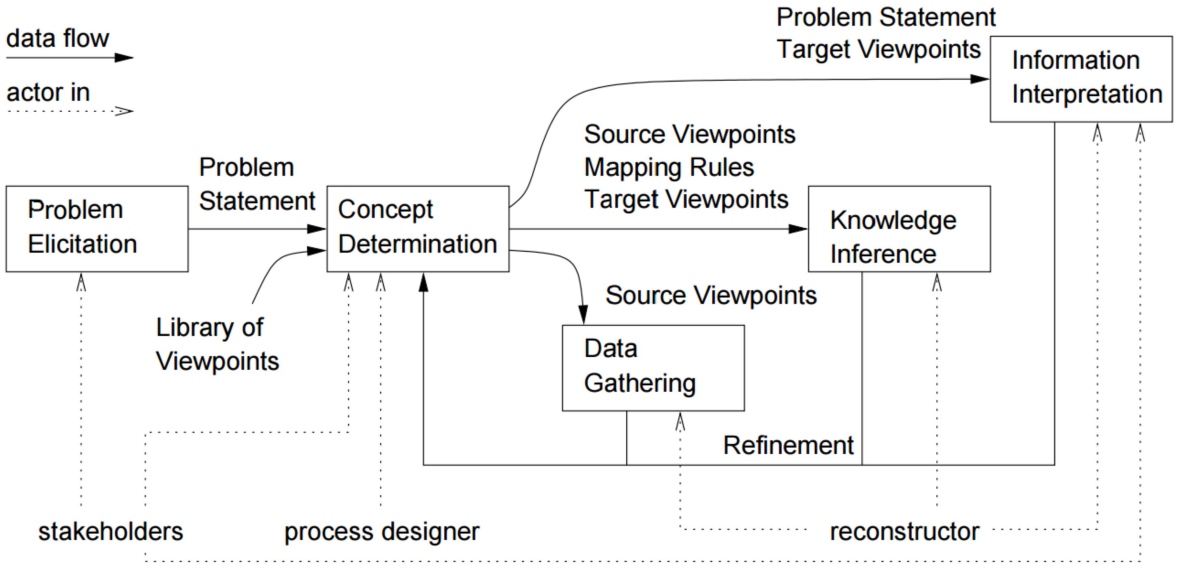
### 2.4.1 Simfonija [1]

Glavni koncept je kombiniranje različnih vidikov in pogledov na sistem. Je metoda, pridobljena na realnih projektih več avtorjev [1].

Pogled opisuje obravnavni sistem, vidik pa pravila in dogovore, ki se uporabljajo za analizo, izdelavo, vizualizacijo pogledov, vsebovanih v vidiku. Vidik torej definira informacije, ki bodo podane v pogledu.

Slika 5 prikazuje tri glavne poglede metode:

- koda (*source view*) – pogled, pridobljen iz izdelkov sistema, kot so izvorna koda, datoteke za prevajanje, datoteke konfiguracije, dokumentacija … Predstavljajo najbolj podroben vidik sistema in največkrat niso del končno pridobljenega modela ravno zaradi prenizkega nivoja abstrakcije;
- želeni pogled (*target view*) – pogled na sistem, ki ga želimo pridobiti na koncu in ki vsebuje informacije, ki bodo pomagale pri reševanju in odpravi problemov. Je preslikava zgoraj opisane kode v višji abstraktni pogled s točno določenimi pravili za preslikovanje, ki so odvisni od obravnavanega primera in se razlikujejo za vsak primer posebej;
- hipotetični pogled (*hypothetical view*) – pogled trenutnega razumevanja sistema, po navadi je pridobljen z intervjuji z vodilnim na projektu/procesu, največkrat ni natančno opredeljen in je večkrat nepravilen (ker člani v nižji strukturi projekta večkrat spreminjajo tehnično implementacijo, ne da bi se vodilni zavedali vseh sprememb).




Slika 5: Simfonija – grafični prikaz korakov [1].

### 2.4.2 Kakofonija [2]

Metoda rekonstrukcije programske opreme na osnovi metamodelov.

Model je poenostavljen prikaz obravnavanega sistema, ki naj bi podal prikaz vseh pomembnih aspektov problemske domene in sistema.

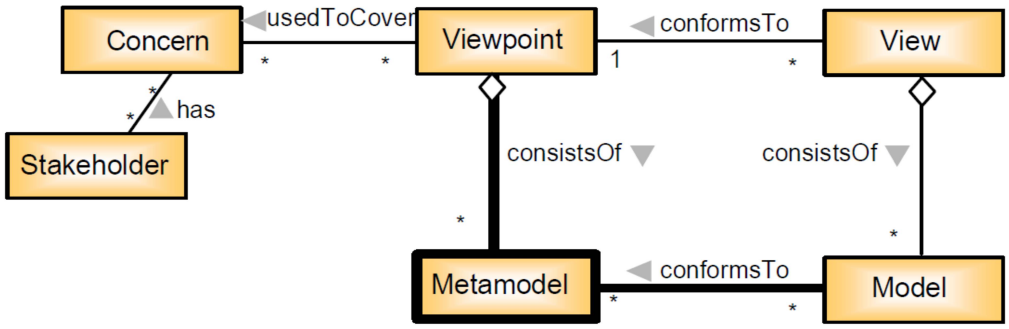
Je nekoliko širši model kot prej opisana simfonija, ker so določeni pogledi zgrajeni iz več modelov.

Kakofonija je ozko povezana tudi s pristopom modelno usmerjenega razvoja (MDE – angl. *Model Driven Engineering*), v katerem metamodeli niso samo narisane škatlice, povezave in puščice, ampak produktivna sredstva, ki nam omogočajo:

- pridobiti precizne in operativno uporabne vidike arhitekture
- povezovanje abstraktnih konceptov z obstoječimi izdelki *metaware*,
- vodenje celotnega procesa rekonstrukcije ne glede na nivo abstrakcije arhitekture, ki jo želimo pridobiti.

Je metodologija, ki ni naravnana na točno izbrano tehnologijo. Poudarek je na dejstvih, ki izhajajo iz programske opreme.

Slika 6 prikazuje model na najvišji ravni, t. i. megamodel, in povezavo z metodo simfonije.




Slika 6: Združitev megamodelov MDE in SA [2].

Pri vsaki rekonstrukciji [2] je pomembno najprej narediti prve tri korake, ki nato definirajo nadaljnji potek rekonstrukcije in nivo abstrakcije, ki ga želimo doseči:

- opredelitev ciljne interesne skupine, kateri je namenjen model, ki bo izdelan z rekonstrukcijo,
- formalizacija pomena modela aplikacije za zgoraj opisano interesno skupino,
- ugotoviti, kateri vidiki so najbolj pomembni za ciljno skupino.

## 2.5 Orodja za avtomatsko izdelavo modela

V devetdesetih letih 20. stoletja je bilo veliko pričakovanj od orodij, ki naj bi omogočala rekonstrukcijo modela in kode. Čeprav so bile vse rešitve nepopolne in niso izpolnile pričakovanj, se še vedno uporabljajo, ker pripomorejo k hitrejši rekonstrukciji arhitekture. V nadaljevanju je podanih nekaj orodij, ki pomagajo razvozlati arhitekturo zapletenih sistemov. Zaradi kompleksnosti orodij so podani samo splošni in kratki opisi.

### 2.5.1 Rigi [4]

Rigi je orodje, ki omogoča ekstrakcijo podatkov iz kode in vizualizacijo (recimo iz kode v jeziku C s pobarvanimi diagrami). Je odprto in fleksibilno orodje, kjer so lahko nove funkcionalnosti dodane z uporabo jezika (*Rigi Command Language*). Repozitorij je sestavljen iz grafov poteka, ki opisujejo entitete programske opreme in njihove povezave ter medsebojno sodelovanje. Za izračun konvergenc, razlik in podobnosti se uporablja formalni model. Pridobljeni model prikazuje razliko med modelom sistema, ki ga vidi inženir, in med realnim modelom sistema. Uporablja lasten format za zapis podatkov (RSF – Rigi Standard Format), ki je postal *de facto* standard za ekstrakcijo informacij iz kode. Format Rigi omogoča preprosto zapisovanje relacij med entitetami (oblika je recimo „relation <entity1> <entity2>“).

### 2.5.2 Reflexion models [4]

Model prikazuje razlike in podobnosti med modelom in pogledom z višjega abstraktnega

vidika in modela kode. Inženir določa in izdela model višje abstrakcije ter preslikavo na kodo. Orodje potem izračuna in pokaže področja, na katerih izdelani abstraktni model dobro pokriva kodo in kjer modela nista kompatibilna oz. je treba izboljšati model. Kot zanimivost lahko napišemo, da se je navedena tehnika uporabljala v podjetju Microsoft pri izdelavi arhitekture aplikacije Excel, ki je sestavljena iz več kot 1,2 milijona vrstic kode, napisane v jeziku C. To je razvijalcem omogočilo boljši pregled nad celotno strukturo in hitrejši razvoj ter načrtovanje nadaljnjih korakov.

### 2.5.3 Software Bookshelf [4]

Je ogrodje, ki omogoča zajem, organizacijo in upravljanje informacij o programski opremi ter razširitev z različnimi drugimi orodji, kot je prej opisano orodje Rigi. Repozitorij tega orodja se lahko izdela tudi z uporabo drugih orodij za obratno inženirstvo. Vse informacije znotraj ogrodja se prenašajo z uporabo protokolov Web. Tehnologija Web omogoča lažjo izdelavo različnih prikazov in navigacijo pri izdelavi modela.

### 2.5.4 Relational Algebra [4]

Je tehnika, ki se uporablja za manipuliranje s podatki znotraj določenega orodja. Največkrat se takšna tehnika uporablja za abstrakcijo podatkov iz relacijskih podatkovnih baz. Orodja Dali in Armin, opisana v nadaljevanju, uporabljajo tehniko poizvedb SQL za definiranje pravil za razvrščanje podatkov po določenih kriterijih. Na takšen način se člani projekta osredotočajo na raven informacij, za katere se zanimajo v obravnavanem času. Omogoča tudi lepši in bolj jasen pogled obravnavanega sistema.

### 2.5.5 Dali [4]

Dali je orodje za analizo arhitekture sistema. Za vizualni prikaz uporablja orodje Rigi in temelji na analizi različnih pogledov na sistem, ki omogočajo opis sistema s statičnega in dinamičnega vidika. Pogledi se sestavljajo iz novih in že obstoječih vidikov s ciljem pridobivanja bolj točnega in širšega vidika. Vsebuje bazo SQL, ki vsebuje različne poglede. Uporablja poizvedbe SQL za definicijo pravil razvrščanja.

### 2.5.6 Armin [3]

Armin (angl. *Architecture Reconstruction and Mining*) je orodje, ki je naslednik orodja Dali in je eno od novejših rešitev, trenutno dostopnih za rekonstrukcijo arhitekture. Izdelali so ga na institutu Carnegie Mellon Software Engineering Institute (SEISM) in v podjetju Robert Bosch. Vsebuje vse funkcionalnosti orodja Dali. Uporablja že prej standardizirane formate, tako da za zapis relacij uporablja format Rigi Standard. Vsebuje podporo tudi za jezik Graphical eXtensible Markup Language (GXML).

### 2.5.7 GreQL [1]

GreQL (angl. *Graph Repository Query Language*) je povrpaševalni jezik za t-grafe (angl. *Tgraphs*), ki temeljijo na jeziku GRAL (objektno orientirani jezik, ki uporablja Z-notacijo). Je zelo podoben klasičnemu jeziku SQL in se ga zato lahko relativno hitro nauči. Temelji na regularnih izrazih poti, ki opisujejo pot v grafu.

Poleg vseh naštetih orodij obstaja še cela množica orodij, ki imajo namen olajšati rekonstrukcijo programske opreme, vendar jih je zaradi prevelikega števila nesmiselno naštevati. Ravno zaradi velikega števila različnih orodij lahko sklepamo, da ni generalno uporabnega orodja, ki bi dejansko samostojno razvozlal zgradbo, še zlasti kompleksnih sistemov.

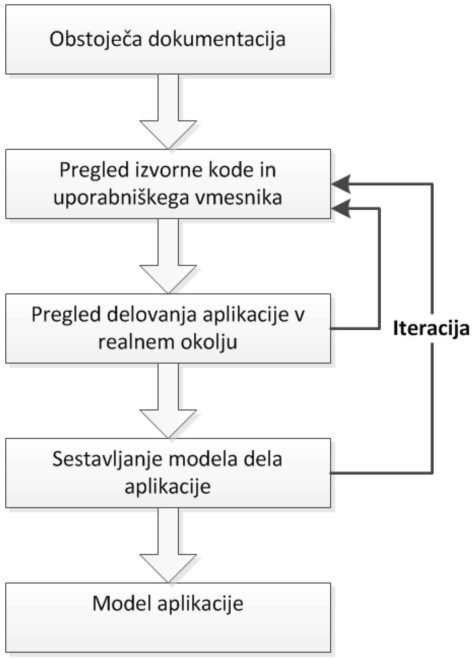
Opisane metodologije in orodja so zaradi kompleksnosti problema samo smernice, kako se lotiti reševanja problema. Vsaka skupina ali posameznik, katerega naloga je izdelati model aplikacije z rekonstrukcijo iz že obstoječega sistema, bo največkrat prilagodila metodologijo in pristop ali uporabila kombinacijo različnih pristopov, metodologij in orodij. Preslikava med kodo in različnimi abstraktnimi modeli je še vedno ostala trd oreh in še vedno ne obstaja nobeno orodje, ki bi razen razrednih diagramov  sestavilo tudi ostale diagrame in različne poglede brez vpliva človeškega faktorja, zato sta v vsakem primeru rekonstrukcije še kako potrebna močan ročni poseg in trdno delo razvijalcev, analitikov in uporabnikov.

# 3 Izdelava modela

Glavno poglavje opisuje proces rekonstrukcije aplikacije in izdelave modela programske opreme iz že delujočega izdelka oz. aplikacije.

Rekonstrukcija aplikacije je proces pridobivanja podrobno dokumentiranega modela iz obstoječe programske rešitve. Pridobljeni model je nekoliko širši in bolj podroben, kot ga po navadi opisujejo v rekonstrukciji arhitekture aplikacije, zato je tudi beseda arhitektura večkrat namenoma izpuščena v besedilu.

Model je sestavljen iz več diagramov, ki opišejo aplikacijo z različnih vidikov (tako statični kot dinamični pogled sistema [6]). Pri vsakem modelu so navedeni grafični prikazi diagramov, njihov jedrnat opis in podroben opis posameznih modelov. Cilj je, da se iz izdelanih modelov in opisov omogoči tudi uporabnikom, ki nimajo izkušenj v obravnavani problemski domeni, razumevanje vseh možnosti aplikacije in vseh procesov.




Slika 7: Potek izdelave modela po fazah.

Slika 7 prikazuje potek izdelave modela po posameznih fazah. Najprej je pregledana obstoječa dokumentacija, potem so v več iteracijah pregledni koda, uporabniški vmesnik in obnašanje aplikacije pri vsakdanji uporabi ter so na koncu sestavljeni različni diagrami. Postopek je potekal z analizo izvorne kode posameznih modulov, razredov in zaslonskih oken. Na koncu je izdelan celoten dokument, dodani pa so tudi opisi posameznih komponent. Obstoječe dokumentacije je bilo zelo malo, zato je prva faza končana hitro in ni imela večjega

pozitivnega vpliva na končni izdelek. V fazi pregleda izvorne kode in uporabniškega vmesnika je vključen tudi pregled funkcioniranja aplikacije v realnem okolju, kjer smo lahko primerjali razumevanje kode z obnašanjem aplikacije.

Pri izdelavi razrednega diagrama je ugotovljeno, da je razredni diagram pravzaprav enak podatkovnemu modelu (če izločimo mejne in kontrolne razrede). To je deloma posledica izbrane metode objektnega programiranja. Dejstvo preslikave podatkovnega modela iz razrednega diagrama v veliki meri poveča transparentnost in razumljivost povezave med entitetnimi tipi in relacijsko bazo podatkov, zato je tudi podatkovni model izpuščen iz podrobnega opisa.

Model aplikacije, ki je pridobljen v procesu rekonstrukcije, je sestavljen iz več različnih diagramov in opisov:

- diagrami primerov uporabe, ki opisujejo vse zmožnosti, ki jih nudi aplikacija, način sodelovanja z zunanjim svetom in uporabniki ter njihove vloge,
- razredni diagrami, iz katerih lažje razumemo koncept celotnega sistema in podatkovni koncept, ki je, kot je že prej omenjeno, v bistvu enak razrednemu diagramu, če izločimo mejne in kontrolne razrede,
- diagrami zaporedja prikazujejo medsebojno komunikacijo in potek informacij najpomembnejših gradnikov aplikacije ter so najbolj namenjeni razvijalcem in vzdrževalcem v prihajajočih časih,
- komponentni diagrami pomagajo pri razumevanju o tem, kako je aplikacija sestavljena in kakšen je pregled glavnih vmesnikov, prek katerih so povezani sestavni deli, podan je tudi diagram možne izboljšave,
- diagrami postavitve podajajo informacijo o različnih verzijah končne postavitve posameznih komponent aplikacije v odvisnosti od zahtev naročnika, geolokacije posameznih profitnih centrov, stopnje izkoriščanja že obstoječe strojne opreme naročnika, zahtev, ki so posledica stopnje obremenitve, itd.,
- diagrami aktivnosti prikazujejo notranje obnašanje dela sistema, prehod stanja, vejitve, odvisne od podatkov, podatkovne tokove in medsebojne odvisnosti procesov,
- časovni diagrami prikazujejo časovno zaporedje korakov določenega procesa ali aktivnosti, iz katerih lahko lažje razumemo ozka grla pri doseganju želene hitrosti in odzivnosti aplikacije.

## *3.1 Primeri uporabe – zmožnosti aplikacije*

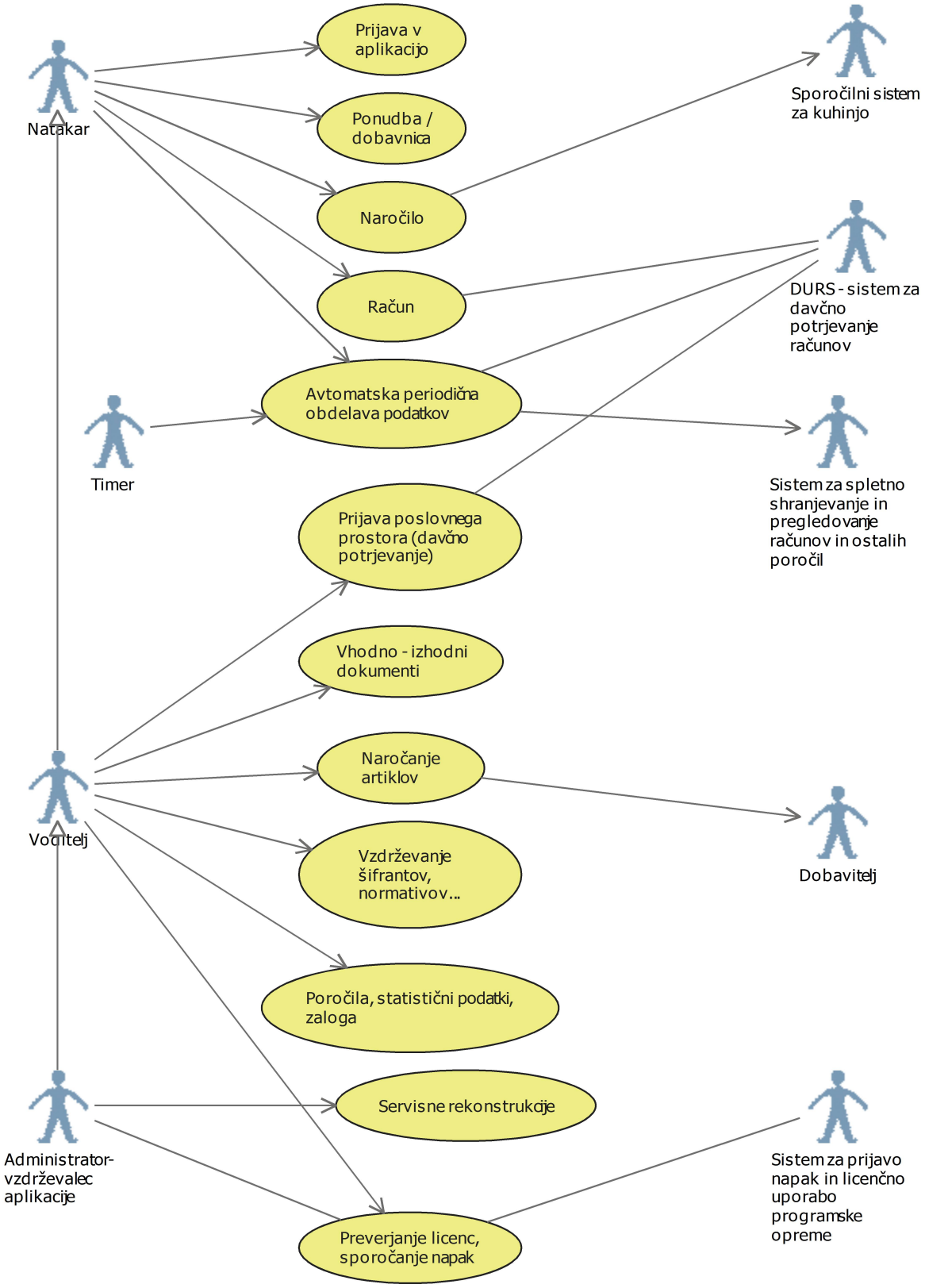
Kot najhitrejši in najbolj razumljiv opis možnosti določene aplikacije se je izkazal diagram primerov uporabe. Z njim lahko relativno hitro opišemo problemsko domeno in možnosti, ki jih trenutno vsebuje programska oprema.

Zaradi zapletenosti same aplikacije in velikega števila možnosti je v glavnem diagramu uporabe, ki je prikazan na sliki 8, združenih nekaj diagramov uporabe, ki so razčlenjeni v nadaljevanju.

### 3.1.1 Diagram primerov uporabe – glavni diagram

Opis posameznih primerov uporabe (opisani so primeri uporabe, ki v nadaljevanju niso razčlenjeni na poddiagrame):

- Prijava v aplikacijo – omogoča zagon aplikacije in identifikacijo uporabnika, v primeru večkratnega napačnega vnosa se aplikacija zaklene, modul preverja tudi datum predplačila aplikacije oz. veljavnost licence, pri prijavi v aplikacijo se preverja tudi pravilen datum in pravilen čas, ki ga aplikacija ponudi iz sistemske ure, poleg tega se tukaj izvede še nekaj akcij (preverjanje periodične izdelave arhiva podatkovne baze, pri posodobitvi aplikacije bo aplikacija opozorila na novosti v programu ali mogoče na nove zakonske predpise, pri prehodu v novo leto bo opozorila na akcije, ki jih mora uporabnik opraviti – zaključek leta in prenos zaloge v novo leto …).
- Ponudba/dobavnica – izdelava ponudbe ali dobavnice v sistemu restavracije (ponudba ne spreminja zaloge, dobavnica pa jo). Ponudbe se po navadi izdelajo v primeru večjih dogodkov (poroke in podobne slovesnosti), kjer potencialni kupci želijo pridobiti ovrednotene ponudbe po lastnih zahtevah – so neobvezujoči dokumenti. Dobavnice se izdajo pogodbenim kupcem večkrat mesečno ali večkrat dnevno, račun se nato izdela periodično za vse odprte dobavnice v izbranem obdobju za izbranega znanega kupca.
- Prijava poslovnega prostora – pred potrjevanjem prvega računa je treba na DURS prijaviti poslovni prostor (pošlje se davčno številko, točen naslov, poštno številko, kraj, številko katastrske občine, številko stavbe in številko dela stavbe). Komunikacija poteka prek varne povezave, podatke se pošlje ovite z ovojnico SOAP in podpisane s certifikatom davčnega zavezanca. Brez uspešne prijave poslovnega prostora ne moremo potrditi računov na obravnavanem prodajnem mestu.




Slika 8: Diagram primerov uporabe celotne aplikacije.

### 3.1.2 Primer uporabe Naročilo

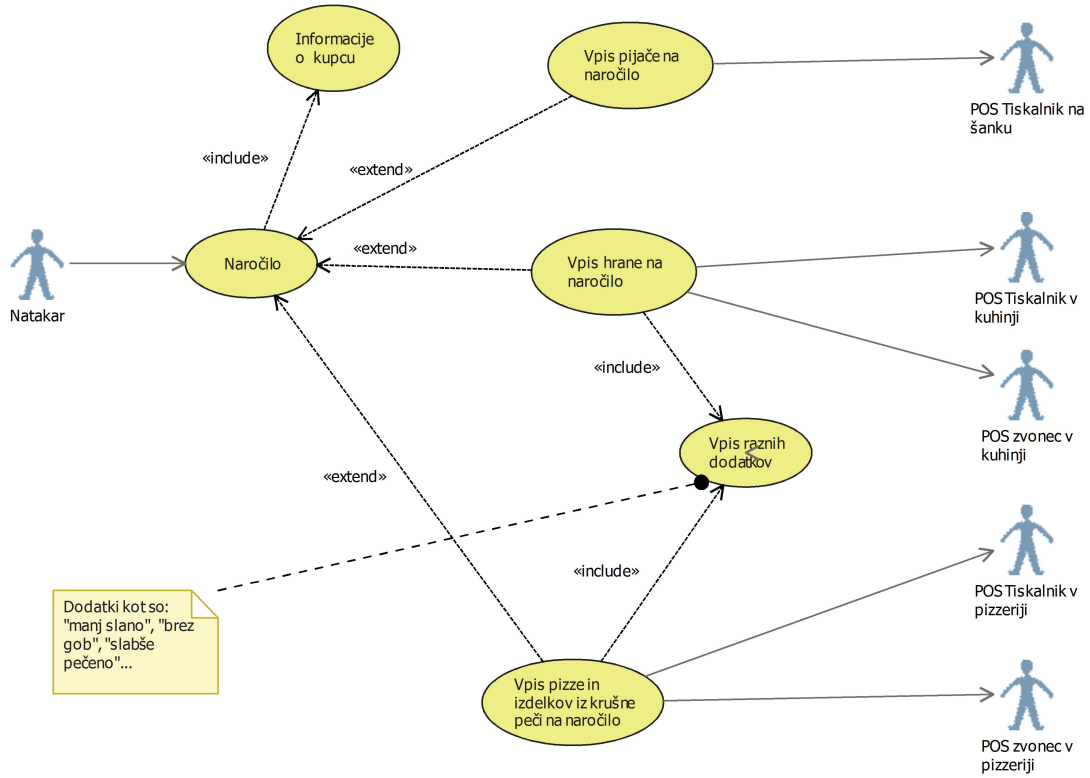
Na sliki spodaj je prikazan primer uporabe Naročilo s podrobno opisanimi vsebovanimi primeri uporabe.

Glavni akter je natakar, ki pri izdelavi naročila vpiše podatke o kupcu. Nato lahko k naročilu doda hrano, pijačo ali tudi tretjo skupino izdelkov (recimo pice). Vsakemu izdelku lahko pridruži tudi poljubne dodatke, ki bolj podrobno opišejo željo stranke in ki jih ne moremo opredeliti na tak način, da jih vpišemo kot izdelek (recimo bolj slano, brez gob, srednje pečeno itd.).

Pri izdelavi naročila je treba definirati tudi kupca ali vsaj mizo oz. prostor, na katerega se naročilo nanaša.

Po zaključku naročila in vpisa v bazo se glede na parametre aplikacije posamezno naročilo natisne na tiskalnike POS na točilnem pultu, kuhinji, oddelku picerije.

Sistem omogoča tudi obveščanje akterjev v kuhinji, da je naročilo prispelo (recimo zvonec ali utripajoče lučke).




Slika 9: Diagram primera uporabe Naročilo – razčlenjen.

### 3.1.3 Primer uporabe Račun – razčlenjen

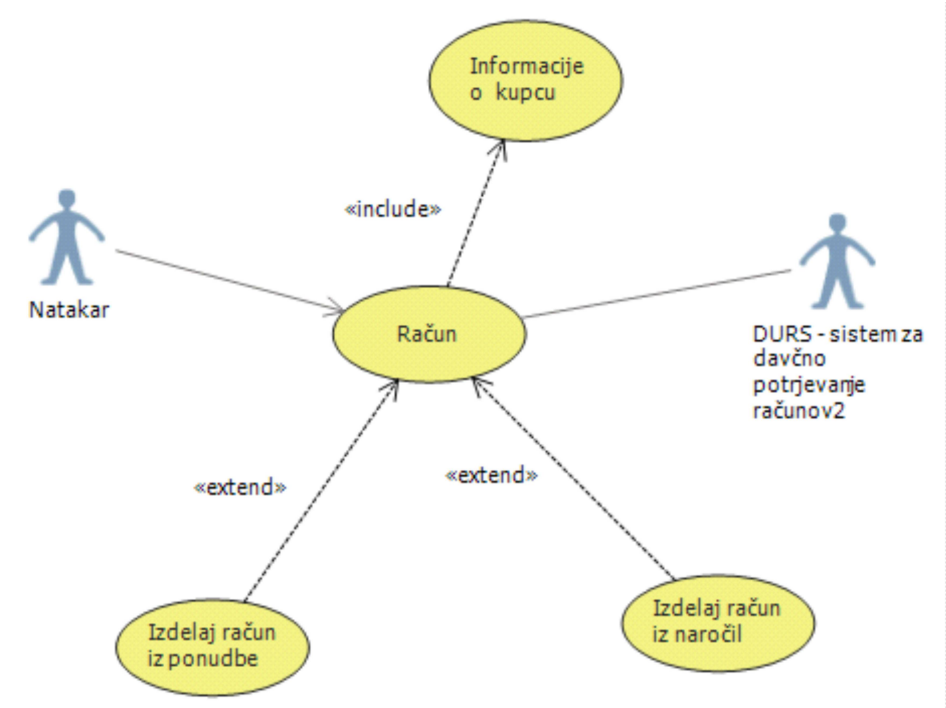
Primer uporabe Račun opisuje izdelavo računa, kjer je glavni akter natakar.

Račun lahko izdelamo na več načinov:

- direktno z dodajanjem izdelkov na zaslonsko masko in z zaključkom računa – če ni bilo predhodno izdelano naročilo,
- iz obstoječega naročila ali iz ponudbe/dobavnice, če so ti podatki predhodno vpisani. Z izdajo računa se v tem primeru povezano naročilo/ponudba/dobavnica tudi zaključi in ga ni več možno spreminjati ali iz njega izdelati novega računa.

V vsakem primeru se mora za  izdelavo računa vnesti tudi podatke o kupcu oz. izbrati kupca (v primeru, da se eksplicitno ne izbere kupca, se prodaja avtomatsko pridruži na maloprodajnega kupca).

Pred izpisom se račun poskusi davčno potrditi na DURS-u. Servis DURS za potrjevanje je tukaj napisan kot zunanji akter, ki mu pošljemo podatke o računu in od katerega pričakujemo odgovor. Če potrjevanje v določenem času ne uspe, se račun po preteku časa izpiše, za potrjevanje pa poskrbi modul  za avtomatsko obdelavo podatkov, ki bo naknadno poskusil potrditi obravnavani račun in vpisati podatke, pridobljene od DURS-a, v bazo ter jih pridružiti obravnavanemu računu.




Slika 10: Diagram primera uporabe Račun – razčlenjen.

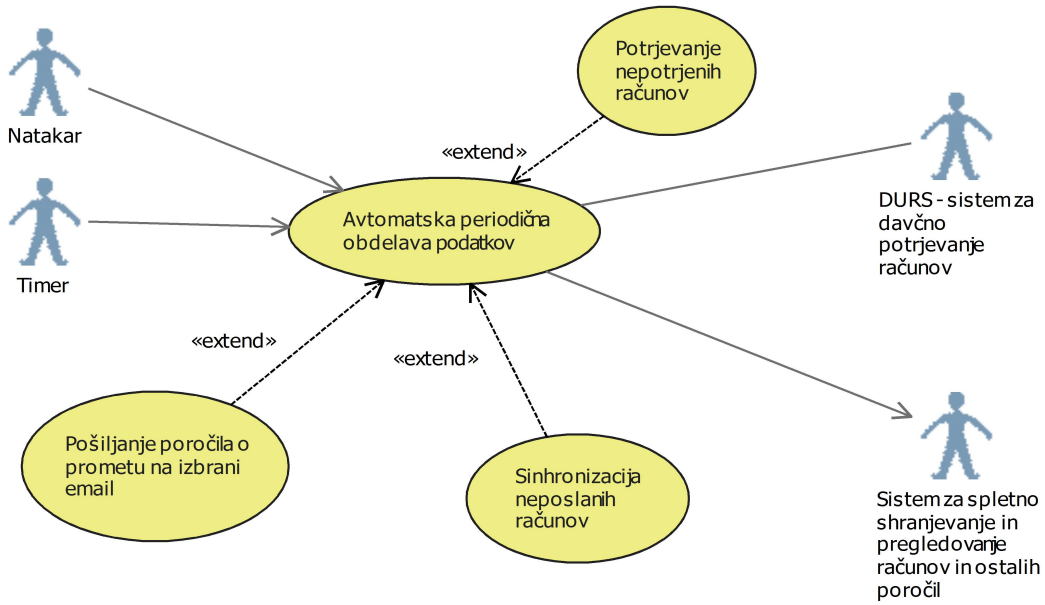
### 3.1.4 Primer uporabe avtomatske periodične obdelave podatkov

Avtomatska periodična obdelava podatkov je proces, ki se periodično zažene v ozadju s pomočjo časovnika ali ročnega posega s strani natakarja.

V tem postopku se pregledajo vsi nepotrjeni računi, ki so primerni za potrjevanje (vsi računi, razen z načinom plačila prek transakcijskega računa). Če se v procesu potrjevanja le-to uspešno izvede, se račun v podatkovni bazi označi kot potrjen in v naslednji iteraciji ne bo več obravnavan.

Druga nit tega procesa pa skrbi za sinhronizacijo podatkov prek spletnih servisov in shrambo v oblaku. Pošiljajo se računi in vsi pripadajoči objekti (postavke računov, artikli, izdelki). Te podatke se nato lahko pregleda s pomočjo spletne aplikacije na več različnih načinov. Če se v procesu sinhronizacije uspešno vpišejo podatki v oblak in če od spletnega servisa dobimo pozitivno potrdilo, se vsi podatki označijo kot sinhronizirani in v naslednji iteraciji ne bodo obravnavani. Podatki se pošiljajo v oblak samo v primeru, da je opcija vklopljena in ima kupec le-to zakupljeno.

Odvisno od nastavitve parametrov aplikacije se pošiljajo tudi določena poročila s prometom objekta na izbrano elektronsko pošto voditelja objekta oz. na več različnih elektronskih naslovov.




Slika 11: Diagram primera uporabe AOP – avtomatska obdelava podatkov – razčlenjen.

### 3.1.5 Primer uporabe vhodno-izhodnih dokumentov

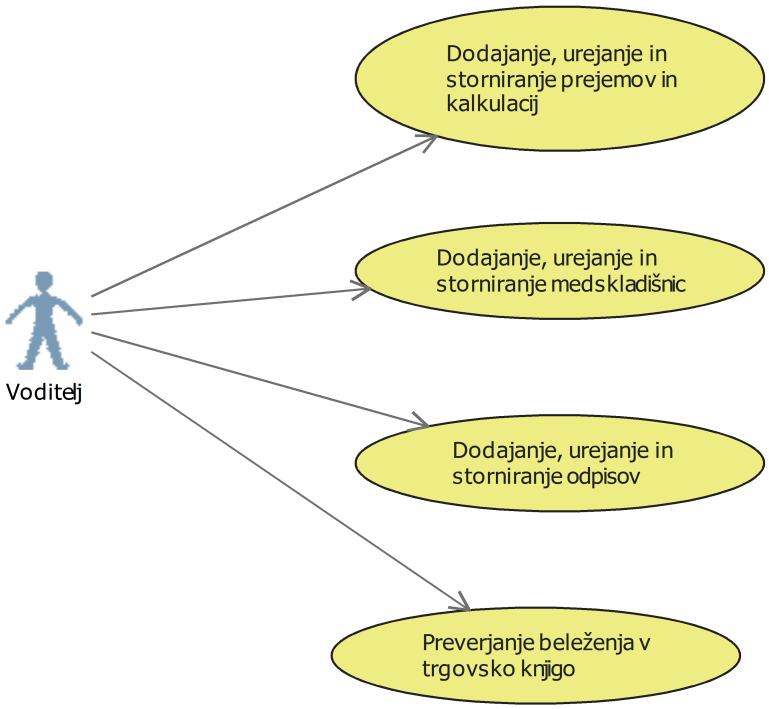
Vhodno-izhodni dokumenti vsebujejo popis artiklov, cen in izračunov pri vhodu na zalogo oz. pri povračilu – izhodu iz zaloge.

Naredimo lahko vnos, urejanje, storno prijeme in pripadajoče izračune. Pri vnosu se dodajo želeni artikli, vpišeta se nabavna in morebitna prodajna cena. Aplikacija pa računa ostale podatke (vmesne vsote, končne vsote, marže, tečaj, če obstaja, ...). Pri prevzemu je treba izbrati izhodiščno (če obstaja) in ciljno skladišče (če obstaja), dobavitelja in vpisati morebitne dodatne stroške (prevoza, carine itd.), ki so nujni za izračun realne marže. Prejem je možno izdelovati v več korakih, ga shraniti in zopet urejati, vse dokler ni zaključen. Z zaključevanjem prejema se vsi podatki obdelajo in povzročijo spremembe v drugih povezanih modulih (povečanje/zmanjšanje zaloge, sprememba trgovske knjige, sprememba cenikov, zapisniki).

Če ima poslovni subjekt več podružnic (profitnih centrov), se lahko določeno blago pošlje iz ene poslovalnice v drugo. V tem primeru se izdela medskladiščnica, s katero se določeno skladišče razdolži, ciljno pa zadolži za določene količinske in finančne vrednosti. Dokument je analogen prejemu.

Odpisi so dokument, s katerim se za določeno količino poškodovanega, starega ali zaradi drugih razlogov neuporabnega blaga razdolži zaloga. Dokument je analogen prejemu.

Še zadnji primer uporabe znotraj tega primera je preverjanje beleženja v trgovsko knjigo artiklov, ki so označeni kot trgovsko blago. Knjiženje se dogaja avtomatsko z zaključevanjem izračuna.




Slika 12: Diagram primera uporabe vhodno-izhodnega dokumenta – razčlenjen.

### 3.1.6 Primer uporabe naročanja artiklov

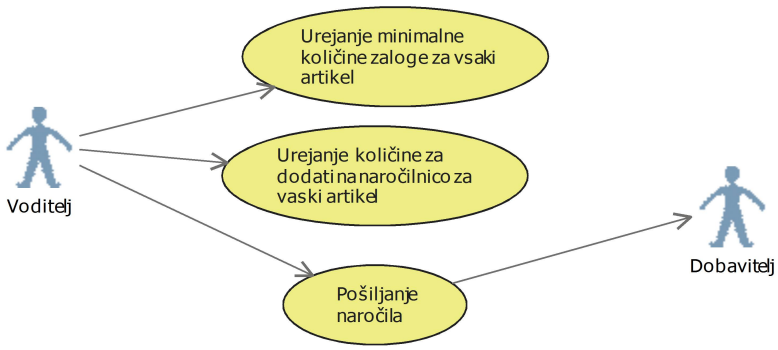
Naročanje artiklov je proces urejanja minimalne količine zaloge, pod katero se blago avtomatsko doda na naročilnico za dobavitelja. Pri vsakem artiklu je treba vpisati količino, ki se bo vpisala na naročilnico, ko zaloga pade pod zgoraj vpisano mejo. Za vsak artikel obstaja tudi možnost izbire, ali se bo sploh vpisal na naročilnico (recimo za artikle, za katere ni smiselno pošiljati naročila oz. jih poslovni subjekt priskrbi samostojno).

Pomembne so tri informacije, ki se urejajo za določen artikel:

- podatek, ali se za določen artikel vodi avtomatsko naročilo,
- minimalna količina zaloge, pri kateri bo artikel avtomatsko dodan na naročilnico,
- količina, ki bo vpisana na naročilnico, če je izpolnjen zgornji pogoj.

Naročilnica se bo spreminjala v odvisnosti od stanja podatkov v podatkovni bazi. To stanje se spreminja z vsakim izdanim računom, izdano dobavnico, vhodno-izhodnim dokumentom, inventuro itd.

Funkcionalnost se po navadi uporablja v večjih poslovnih subjektih, kjer voditelj ne more fizično nadzorovati stanja zaloge. V manjših objektih s samo nekaj zaposlenimi funkcionalnost ni koristna, ker imajo vsi zaposleni dober pregled nad zalogo in lahko pravočasno ukrepajo ter ročno sestavljajo naročila.



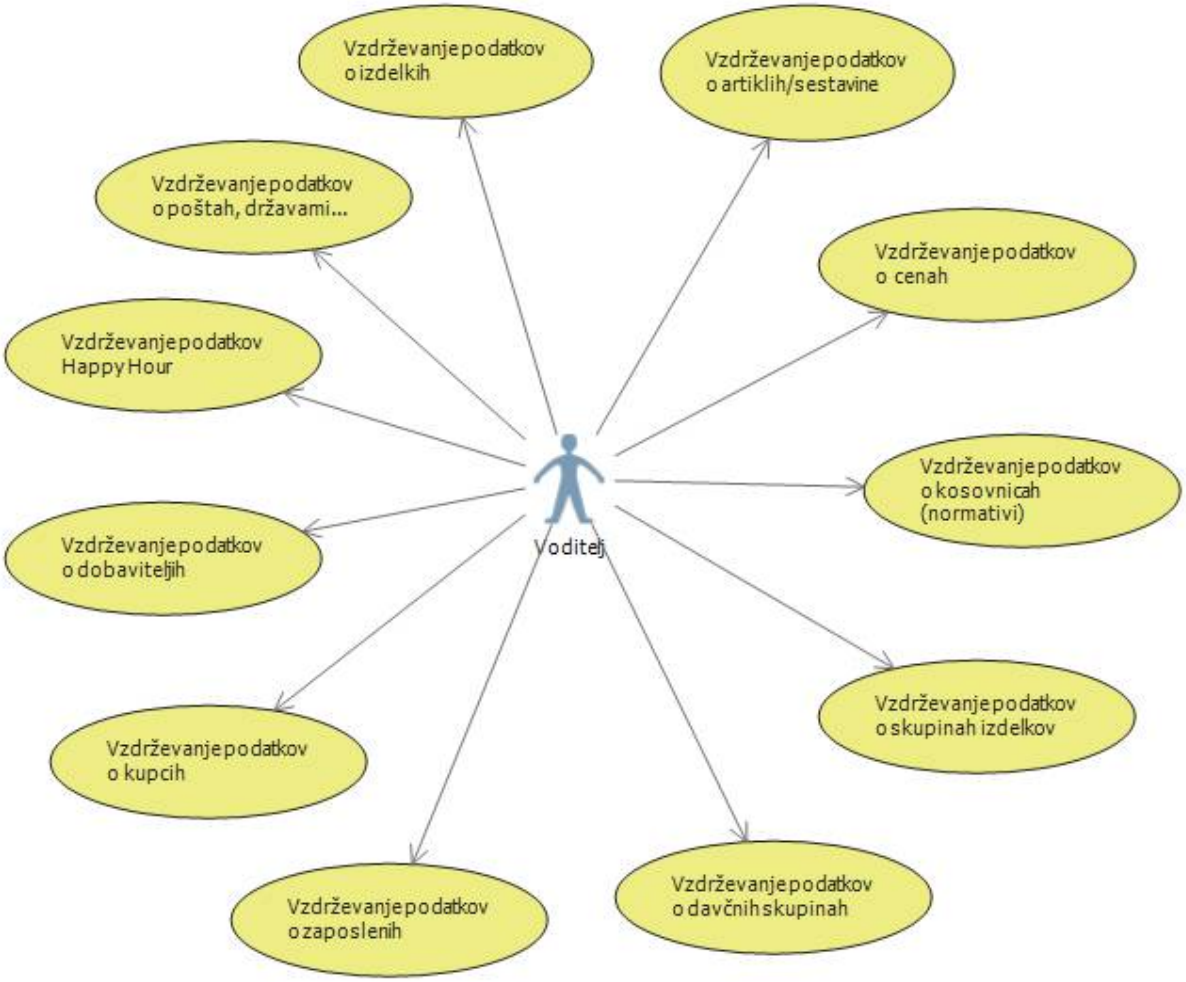

Slika 13: Diagram primera uporabe izdelave naročilnice za dobavitelja – razčlenjen.

### 3.1.7 Primer uporabe vzdrževanja šifrantov, normativov ...

V tem primeru so razčlenjeni različni primeri uporabe. Glavni akter je voditelj.

Prvi primer se nanaša na vpis, urejanje in vzdrževanje podatkov o izdelkih (naziv, pripadnost davčni skupini, enota mere). Izdelki so v gostinstvu lahko sestavljeni iz nekaj artiklov/sestavin – recimo izdelek „Kava z mlekom“ je sestavljena iz 0,007 kg artikla „Kava“, 0,005 kg artikla „Sladkor“ in 0,05 l artikla „Mleko“.

Vnos/urejanje sestavin je eden od primerov uporabe, ki povezuje artikle in izdelke ter medsebojna razmerja v količinah. Vsakemu izdelku je treba določiti tudi ceno, ki ni nujno vezana na ceno sestavin. Skupine izdelkov omogočajo izpis naročil kupcev na različnih lokacijah, ker lahko izdelke razdelimo na pijačo, hrano itd. V primeru uporabe davčne skupine vnesemo znesek davka za določeno skupino oz. oprostitev davka za določene skupine. Davčne skupine potem pridružimo posameznemu izdelku. Če se recimo davek zamenja, je dovolj, da spremenimo davek le v davčni skupini, ne pa tudi pri vsakem izdelku posebej. Z vzdrževanjem ostalih šifrantov (Kupci, Zaposleni, Dobavitelji) urejamo nazive, naslove, davčne številke posameznih objektov. Modul HappyHour določa spremembe cen posameznih ali vseh artiklov v določenem časovnem obdobju (možna je izbira časovnega intervala v dnevu, datum začetka in konca, dneva v tednu ...).




Slika 14: Diagram primerov uporabe vzdrževanja šifrantov, normativov ... – razčlenjen.

### 3.1.8 Primer uporabe poročila, statističnih pregledov in zaloge

Spodnja slika prikazuje diagram primerov uporabe poročila, statističnih pregledov in zaloge.

Aplikacija vsebuje več različnih poročil in pregledov.

Prvi je pregled zaloge oz. trenutnega količinskega stanja artiklov in analogni pregled zaloge za pretekla obdobja. Zaloga je ovrednotena tudi finančno, če so ti podatki vpisani v prejemih. Na pregledu zaloge je lahko razvidno:

- količina in nazivi artiklov, enote mere,
- nabavna cena (če obstaja), skupna nabavna cena za količino,
- prodajna cena (če obstaja), skupna prodajna cena za količino,
- vsote vseh cen.

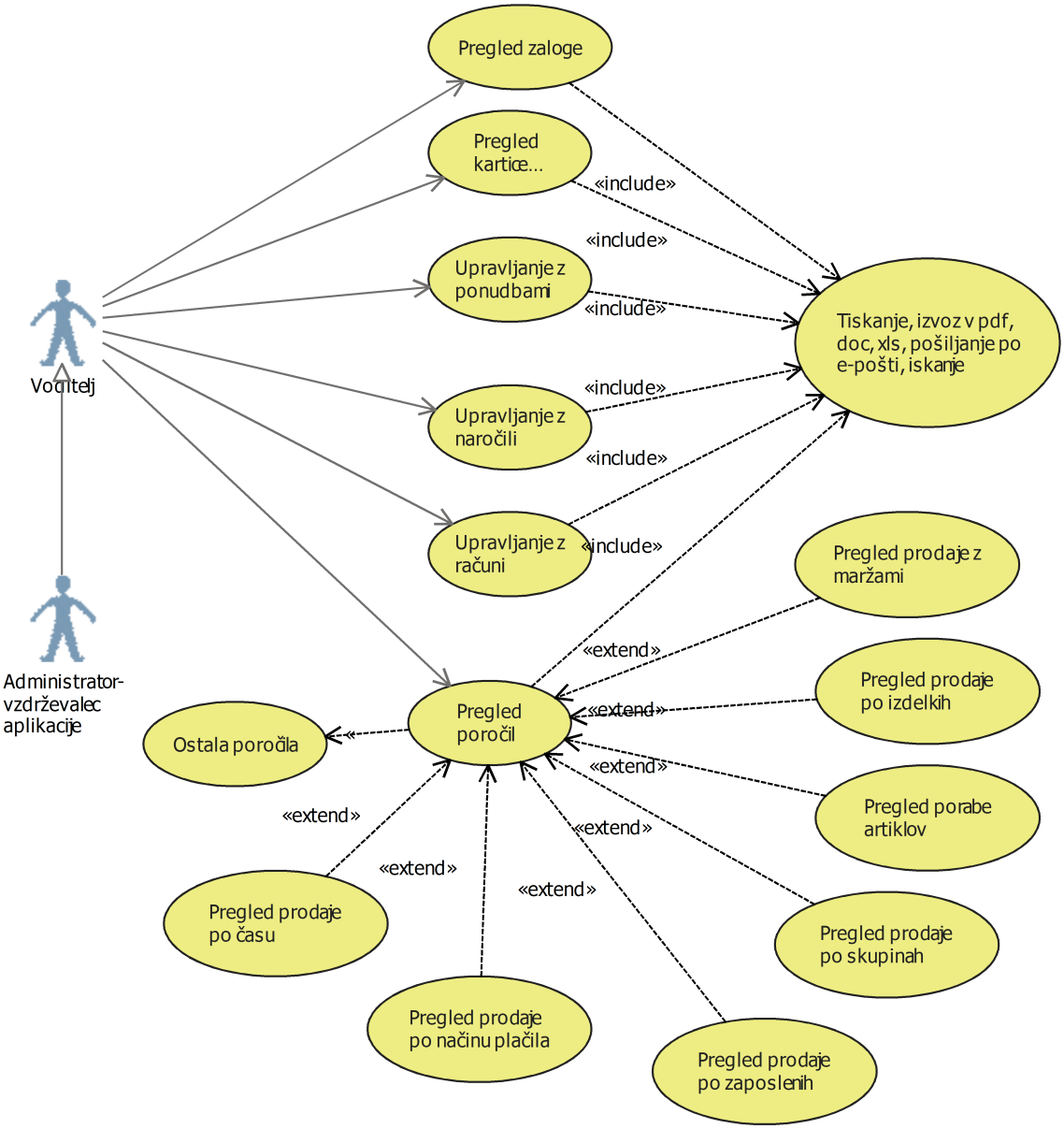
Pregled kartice artikla določenega artikla pomeni pregled časovnega spreminjanja zaloge za obravnavani artikel (za vsako spremembo se dnevno beleži vhod, izhod, prodaja, prenos, povračilo ...).

Drugi primer se nanaša na upravljanje z že izdanimi ponudbami, naročili, dobavnicami in računi. Že izdelane dokumente lahko ponovno natisnemo, izvozimo v različne oblike ali pošljemo po elektronski pošti. Število naknadnih izpisov kopij računov se beleži in izpisuje pri izpisu.

Pregled poročil prodaje obsega več različnih poročil:

- recimo z maržami, kjer se za vsak izdelek izračuna vrednost sestavin v izbranem obdobju in na osnovi prodajne cene računa tudi zaslužek, tj. marža,
- poročilo porabe, kjer se izračuna poraba artiklov glede na vpisane kosovnice,
- prodaja po načinu plačila računov, kjer so zneski skupne prodaje razdeljeni za vsak način plačila posamezno (recimo gotovina: 250,45 EUR, Mastercard 789,00 EUR),
- poročilo prodaje po zaposlenih, kjer je razvidna razporeditev prometa po natakarjih,
- pregled prodaje po času ali po dnevih (s pregledom po času oz. po urah lahko voditelj ugotovi, kdaj je smiselno povečati število delavcev v objektu).

Vsa našteta poročila lahko pregledamo za želeno obdobje, lahko si jih ogledamo na ekranu, natisnemo ali izvozimo (v formatu pdf, doc, xls) ...




Slika 15: Diagram primera uporabe poročila, statistični pregledi, zaloga – razčlenjen.

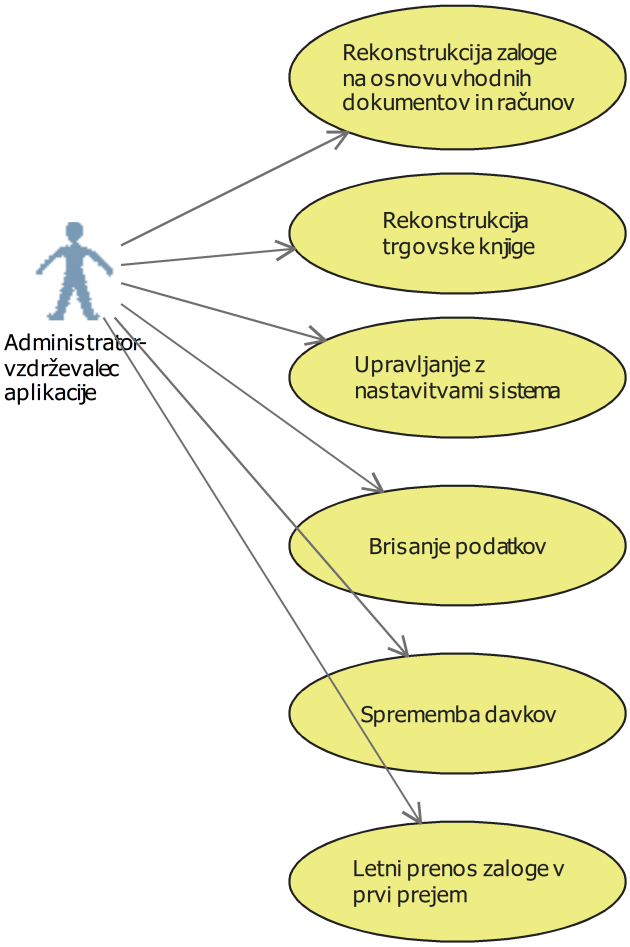
### 3.1.9 Primer uporabe servisne rekonstrukcije

Servisne  rekonstrukcije – modul, namenjen izkušenim voditeljem in vzdrževalcem.

Omogoča brisanje določenih podatkov in rekonstrukcije zaloge, trgovske knjige, kartice artiklov na podlagi računov, dobavnic, vhodno-izhodnih dokumentov, uporablja se največkrat pri uvozu podatkov iz drugih baz in pri prehodu na nov sistem, kadar je treba na osnovi uvoženih podatkov narediti začetno stanje povezanih objektov (zaloge, trgovske knjige ...).

Uporablja se tudi v primeru, ko se aplikacija določeno obdobje testira pri kupcu z realnimi podatki (razen potrjevanja) in ne želimo po zagonu produkcije ponovno vpisovati vseh šifrantov, cenikov itd.

Uporablja se tudi pri zaključku leta, tj. pri letnem prenosu zaloge v prvi prejem v letu. Opcija se uporablja, kadar se ne dela klasična inventura na koncu leta, ampak se kot realno stanje vzame zaloga na koncu leta.




Slika 16: Diagram primera uporabe servisne rekonstrukcije – razčlenjen.

### 3.1.10 Primer uporabe preverjanja licenc in napak

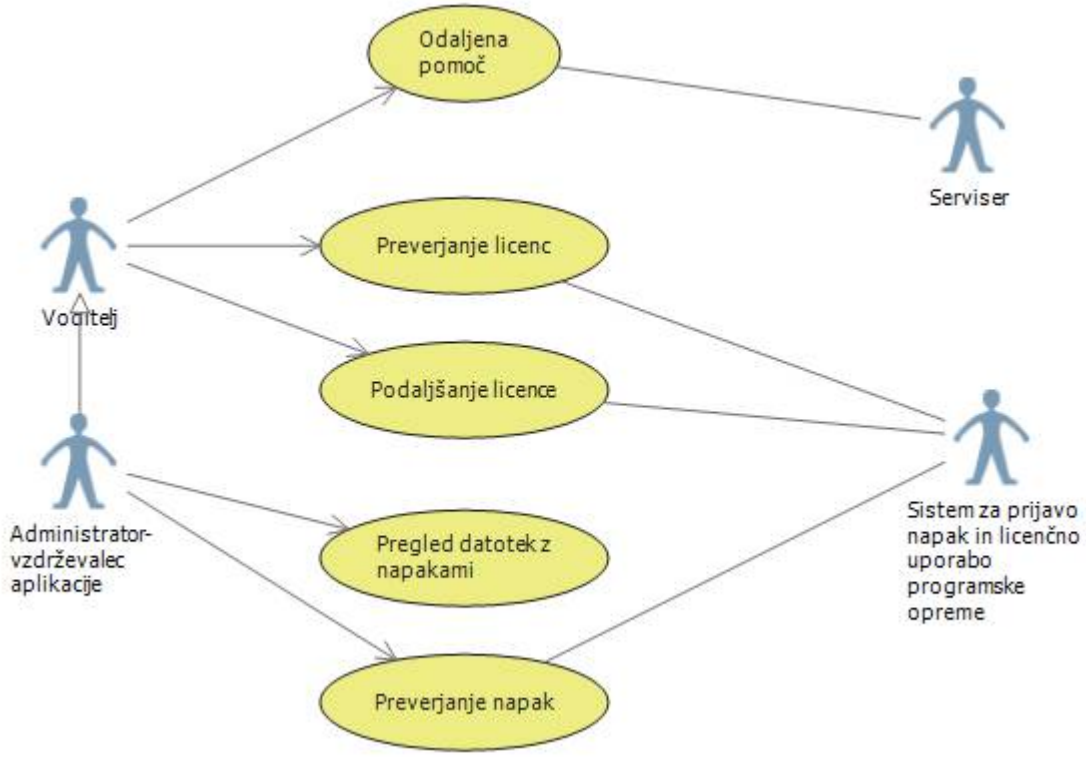
Administrator sistema ali voditelj lahko pri reševanju problemov zažene modul za oddaljeno pomoč. Modul naključno generira 9-mestno število, katerega akter sporoči vzdrževalcem ali podpori in prek katerega se lahko osebje iz podpore poveže na problematično delovno postajo. Postopek je podoben kot pri oddaljenem namizju, le da v tem primeru lahko oddaljena pomoč poteka mimo usmerjevalnikov in požarnega zida.

Preverjanje licenc pokaže podatek, do kdaj je licenca aplikacije aktivna oz. do kdaj je treba urediti podaljšanje.

Podaljšanje licence je proces vpisa licenčnega ključa in s tem podaljšanja veljavnosti licence.

Administrator lahko pri reševanju problemov pregleda tudi datoteke, v katere se vpisujejo razni podatki – tudi napake. Če iz pregledanih datotek ne more razbrati vzroka, potem datoteke posreduje vzdrževalcem.

Sistem za sporočanje napak temelji na pošiljanju informacije o napakah razvijalcem v realnem času, kadar pride do napake. V tem primeru se ne pošiljajo zaupne informacije, ampak samo vhodni parametri funkcije, kjer je prišlo do napake, in opis napake.

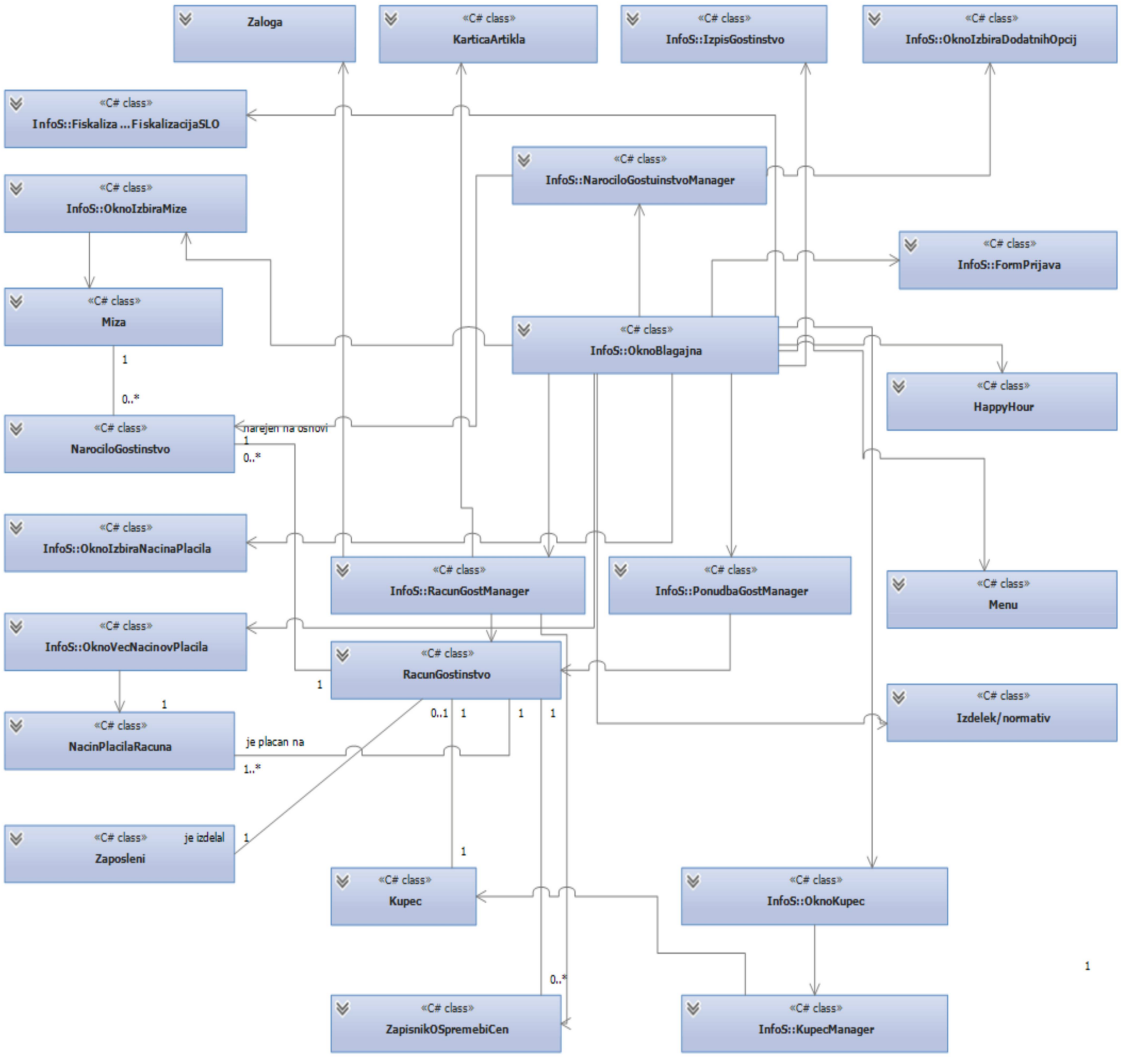



Slika 17: Diagram primera uporabe preverjanja licenc in napak – razčlenjen.

## 3.2 Razredni diagrami

Razredni diagrami so zaradi velika števila entitetnih tipov, kontrolnih in mejnih razredov razdeljeni v več diagramov. Celoten diagram obsega čez 200 razredov in bi bil nepregleden. Slika 18 prikazuje razredni diagram najpomembnejših entitetnih diagramov, kontrolnih razredov in mejnih razredov. Atributi in operacije so zaradi preglednosti izpuščene in so prikazane v tabeli 1.

Pri zagonu aplikacije se po začetnih morebitnih posodobitvah baze (če so te potrebne glede na novo verzijo različice) preberejo parametrizirani podatki iz baze, ki so odvisni od kupca oz. nastavitev. Potem se uporabniku prikaže okno za prijavo, kjer mora vpisati uporabniško ime in geslo. Če je bil vnos pravilen oz. če uporabnik obstaja v podatkovni bazi, se nadaljuje do okna blagajna. Preden se pokaže glavno okno, bo aplikacija preverila še nekaj pogojev (ali je treba narediti arhiv podatkovne baze, opozorilo o trenutnem datumu in času računalnika, licenco).




Slika 18: Razredni diagram entitetnih tipov, kontrolni in mejni razredi osnovnega okna.

Na oknu blagajne lahko uporabnik na več načinov dodaja izdelke v tabelo izdelkov. To se lahko naredi s klikom na gumb izdelka, z iskanjem po šifri ali nazivu, z vpisovanjem točne šifre ali z bralnikom črtne kode. Razred „oknoBlagajna“ tukaj predstavlja glavni, mejni razred med uporabnikom in aplikacijo. Poleg funkcionalnosti, ki jih vsebuje, pa uporablja tudi več drugih kontrolnih razredov in zaslonskih oken.

Ko je uporabnik končal z dodajanjem artiklov, lahko izbere med več možnostmi. Če je treba narediti račun na znanega kupca, se prek „oknoBlagajna“ pokliče zaslonsko „oknoKupec“, ki omogoči izbiro kupca. Potem uporabnik izbere način plačila tako, da prek „oknoIzbiraNacinaPlacila“ izbere način plačila. S tem je račun zaključen in se prek kontrolnega razreda „RacunGostManager“ naredi objekt, ki pripada entitetnemu razredu „RacunGostinstvo“ s pripadajočim postavkami, in vpis v podatkovno bazo.

Po vpisu v bazo „oknoBlagajna“ pokliče kontrolni razred za davčno potrjevanje, kjer poskusi potrditi račun.

Na koncu se račun izpiše na list formata A4 ali na tiskalnik POS prek razreda „IzpisGostinstvo“.

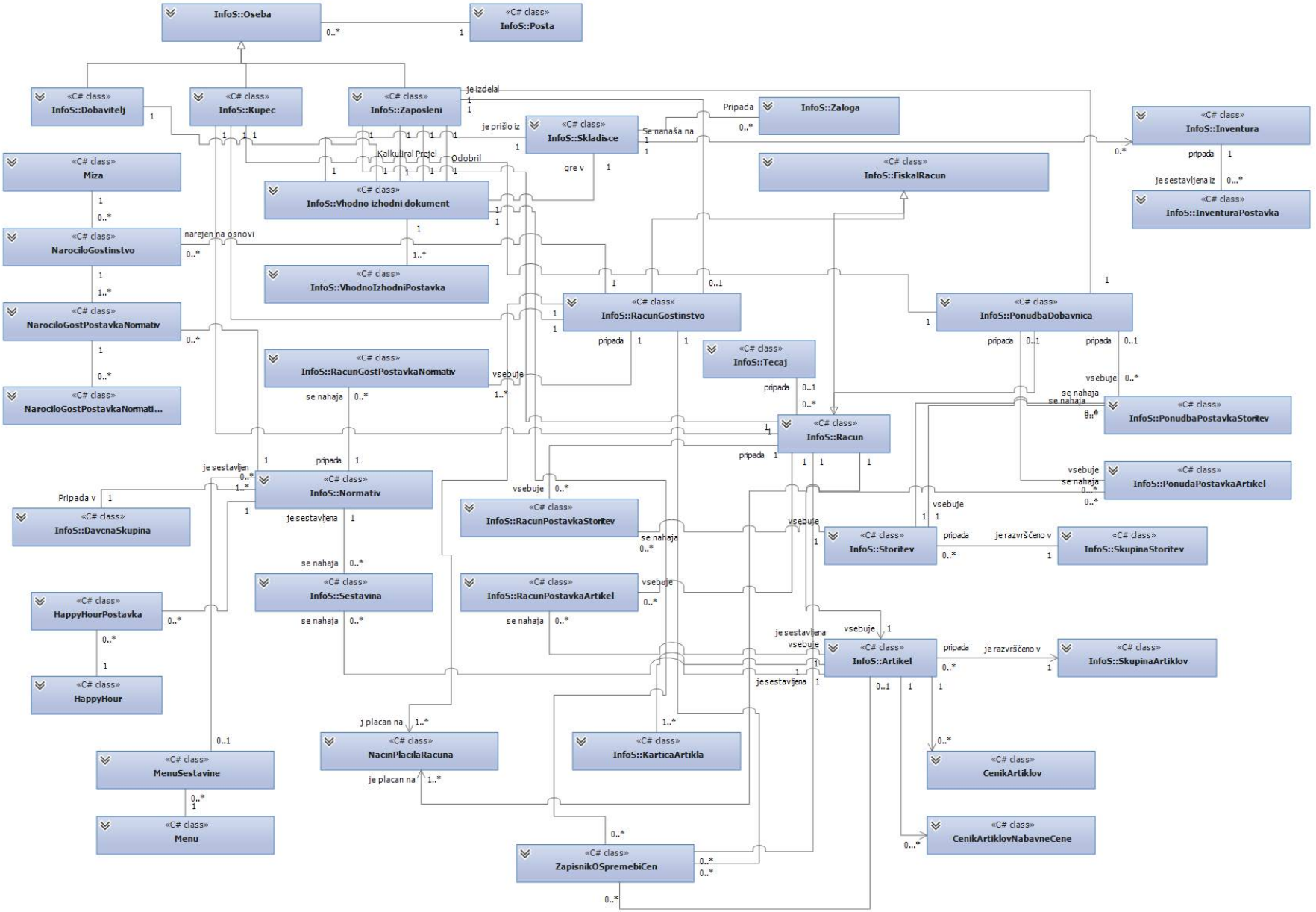
Pri dodajanju izdelkov je mogoče prek blagajne izvesti še več različnih akcij. To so dodajanje popustov prek zaslonske maske „oknoIzbiraDodatnihOpcij“, ki nudi več različnih opcij (popust za celoten račun, storno računa, izpis ponudbe, izpis dobavnice, vnos napitnine, izdelava računa z oprostitvijo DDV-ja ...). Odvisno od izbrane opcije bo „oknoBlagajna“ prek kontrolnih razredov sprožilo nadaljnjo akcijo.

Prek osnovnega zaslona „oknoBlagajna“ je možno izdelati tudi naročilo. Postopek je enak kot pri izdelavi računa, le da ne izberemo načina plačila, temveč kliknemo na gumb za izbiro mize. Prek zaslonskega okna „oknoIzbiraMize“ izberemo mizo, za katero se kupci nahajajo, in izdelamo naročilo. „oknoBlagajna“ bo avtomatsko prek razreda „IzpisGostinstvo“ izpisalo naročilo na enega ali več tiskalnikov POS (odvisno od skupine izdelkov, ki se izpisujejo).

Pri izdelavi računa bo kontrolni razred „RacunGostManager“ poskrbel za spremembo zaloge in kartice artikla. Če gre za trgovske artikle, bo poskrbel tudi za posodobitev zapisov v trgovski knjigi. Celoten proces vpisa računa in postavk v bazo se obdeluje v eni transakciji, kar omogoča konsistentnost podatkov. V primeru, da katerakoli od akcij ne uspe (vpis računa v bazo, vpis postavk, sprememba zaloge, sprememba kartice artikla, sprememba trgovske knjige, izdelava zapisnika), se naredi *rollback* in transakcija se razveljavi.

Slika 19 prikazuje razredni diagram entitetnih razredov in njihova medsebojna razmerja. Diagram je enak tudi modelu podatkovne baze, ker smo pri izdelavi uporabili metodo objektno orientiranega razvoja.

V tabeli 1 je podan seznam vseh razredov (entitetnih, kontrolnih, mejnih, pomožnih ...). Pri opisih so opisana tudi razmerja do drugih razredov in glavni atributi.




Slika 19: Razredni diagram – entitetni tipi.

| Z. št. | Naziv | Opis |
|---|---|---|
| 1 | Oseba | Vsebuje informacije o določeni osebi (ime, priimek, naziv, naslov, kontakt …). |
| 2 | Zaposleni | Naslednik razreda Oseba, dodatno vsebuje še podatke za prijavo v sistem (uporabniško ime, geslo …). |
| 3 | Kupec | Naslednik razreda Oseba, dodatno vsebuje še podatke o skupini kupca (fizična ali pravna oseba), kontakt osebe, davčno številko ... |
| 4 | Dobavitelj | Naslednik razreda Oseba, dodatno vsebuje še nekaj podatkov: kontakt osebo, davčno številko ... |
| 5 | Posta | Šifrant poštnih številk, mest, naselij in držav. |
| 6 | Skladisce | Šifrant skladišč, sestavljen iz identifikatorja in naziva skladišča. |
| 7 | Zaloga | Vsebuje podatke o trenutni zalogi za določen artikel v skladišču, kjer se nahaja. |
| 8 | VhodnoIzhodniDokument | To so prejemi, odpisi, medskladiščnice in podobni dokumenti. Vsebuje informacije o dobavitelju, izdelovalcu, izvornem in ciljnem skladišču, vsotah cen artiklov, ki se nahajajo na dokumentu, dodatnih stroških (prevoz, carina ...), tečaju. |
| 9 | VhodnoIzhodniPostavka | Postavke objekta VhodnoIzhodniDokument. Vsebuje informacije o artiklu, količini, nabavni in prodajni ceni ter marži. VhodnoIzhodniDokument ima lahko 0 ali več postavk. |
| 10 | RacunFiskal | Vsebuje podatek o davčnem potrjevanju (datum potrjevanja, EOR, ZKP, oznako naknadne dostave na potrjevanje ...). Je starševski razred. |
| 11 | RacunGostinstvo | Naslednik objekta RacunFiskal. Vsebuje običajne podatke o računu, kot so informacije o kupcu, izdelovalcu, vsoti računa, načinu plačila, valuti plačila, datumu izpisa, datumu zapadlosti ... |
| 12 | RacunPostavkaNormativ | Postavke objekta RacunGostinstvo. Vsebuje podatke o izdelku, količini, ceni, popustu, davčni stopnji in končni ceni. Objektu RacunGostinstvo lahko pripada ena ali več postavk. |
| 13 | Normativ | Izdelek, ki ga prodajamo v gostinstvu. Vsebuje informacije o identifikatorju, nazivu, ceni, pripadnosti davčni skupini. |
| 14 | Sestavina | Sestavina pripada določenemu normativu/izdelku, na drugi strani pa je povezana z artiklom, ki sestavlja normativ in količine (recimo izdelek „Kava z mlekom“ je sestavljen iz 0,007 kg artikla „Kava“, 0,005 kg artikla „Sladkor“ in 0,05 l artikla „Mleko“. |
| 15 | Artikel | Vsebuje informacije o identifikatorju, nazivu, davku, skupini izdelka, sliki in ceni (če gre za trgovski artikel). Artikli so objekti, ki so povezani z zalogo. |

| | | |
|---|---|---|
| 16 | NarociloGostinstvo | Vsebuje podatke o naročilu kupca (miza, datum naročila, izdelovalec). |
| 17 | NarociloPostavkaNormativ | Postavke objekta NarociloGostinstvo. Vsebuje podatke o izdelku, količini. |
| 18 | NarociloPostavkaNormativDodatek | Morebitni dodatki v prejšnji vrstici, opisane postavke, ki jih ne moremo opredeliti kot končni izdelek. |
| 19 | Miza | Podatek o identifikatorju mize in nazivu oz. oznaki v restavraciji. Je povezan z naročilom. |
| 20 | DavcnaSkupina | Vsebuje podatke od identifikatorju, nazivu, stopnji DDV-ja, stopnji dodatnega davka (če obstaja) oz. podatke o oprostitvi. Davčni skupini pripada eden ali več Izdelkov/Normativov. |
| 21 | HappyHour | Objekt, ki vsebuje podatke o obdobju spremembe cen, dnevih v tednu in urah oz. času, v katerem bodo spremembe cen veljavne. |
| 22 | HappyHourPostavke | Postavke v prejšnji vrstici opisanega objekta. Vsebuje podatke o izdelku in spremenjeni ceni izdelka. |
| 23 | Menu | Objekt, ki je nadgraditev objekta Normativ (izdelek). Meni je sestavljen iz enega ali več izdelkov. Cena je določena samostojno in ni odvisna od vsote cen izdelka, ki ga sestavljajo. |
| 24 | MenuPostavka | Postavke v prejšnji vrstici opisanega objekta, ki vsebuje podatek o izdelku in količini izdelka. Uporablja se za izračun porabe sestavine (artiklov). |
| 25 | NacinPlacila | Podatki o načinu plačila posameznega računa. Obstaja vsaj en podatek za vsak račun. Če je račun plačan na več različnih načinov, potem vsebuje več zapisov: podatek o računu, načinu plačila in znesku plačila. Vsota zneskov načina plačila za posamezen račun mora biti enaka znesku računa. |
| 26 | KarticaArtikla | Vsebuje časovni zapis sprememb zaloge za posamezni artikel. Beleži se vsaka sprememba (vhod, prenos, povračilo, prodaja) za vsak dan in posamezni artikel. Končno stanje za določen datum mora biti enako zalogi na opazovani datum. |
| 27 | SkupinaArtikla | Artikle lahko pridružimo skupinam, ki se jih nato uporablja za razvrščanje pri različnih pregledih. Sestavljen je iz identifikatorja in naziva skupine. Artikel lahko pripada le eni skupini. Skupini artiklov lahko pripada 0 ali več artiklov. |
| 28 | CenikArtiklov | Cenik artiklov. Vsebuje podatek o artiklu (identifikator artikla), datum cenika in neto ceno. Vsebuje zgodovino sprememb cen artiklov. |
| 29 | CenikArtiklovNabavneCene | Nabavne cene artiklov se vodijo po ponderirani ceni, tako da vsebuje samo identifikator artikla in neto nabavno ceno. |
| 30 | ZapisnikOSpremebiCen | Zapisnik o spremembi cen se izdela v primeru, kadar določenemu artiklu spremenimo ceno, na zalogi pa obstaja količina, različna 0, z drugačno ceno. V tem primeru je treba |

| | | |
|---|---|---|
| | | zaradi usklajevanja finančne vrednosti zaloge in trgovske knjige sestaviti zapisnik ter spremeniti tudi ceno starih artiklov, ki so že bili na zalogi. |
| 31 | TrgovskaKnjiga | Finančna vrednost zaloge trgovskih artiklov. Vsebuje podatke o zadolžitvi zaloge, spremembah cen (v prejšnji vrstici opisan objekt), razdolžitvi zaloge z računi, vse ločeno po dnevih ... |
| 32 | Storitev | Storitve so ločene od artiklov. Prodaja storitve ne vpliva na spremembe zaloge, kartice artiklov in trgovske knjige. Sestavljena je iz identifikatorja, naziva (opisa), davčne stopnje in cene ter pripadnosti skupini storitev. |
| 33 | SkupinaStoritev | Storitve lahko pridružimo skupinam, ki se nato lahko uporabljajo za razvrščanje pri različnih pregledih. Sestavljen je iz identifikatorja in naziva skupine. |
| 34 | Racun | Uporablja se v modulu maloprodaje. V modulu gostinstva se ne uporablja. Podeduje atribute od starša FiskalRacun, ki vsebuje informacije o davčnem potrjevanju, sam pa vsebuje podatke: identifikator računa, zaporedna številka, leto, datum izdelave, datum zapadlosti, način plačila, vsota artiklov, vsota storitev, skupna vsota, datum izvajanja storitev oz. dobave artiklov. |
| 35 | RacunPostavkaArtikel | Postavke Racuna, kjer so za vsak artikel opredeljeni količina, cena, popust, skupni zneski in davčna stopnja oz. morebitna osvoboditev davka. |
| 36 | RacunPostavkaStoritev | Postavke Racuna, kjer so za vsako storitev opredeljeni količina, cena, popust, skupni zneski in davčna stopnja oz. morebitna osvoboditev davka. |
| 37 | PonudbaDobavnica | Objekt, podoben objektu Racun, samo da se nanaša na ostale skupine (ponudbe, dobavnice, delovne naloge ...). |
| 38 | PonudbaPostavkaArtikel | Postavke PonudbaDobavnica, kjer so za vsak artikel opredeljeni količina, cena, popust, skupni zneski in davčna stopnja oz. morebitna osvoboditev davka. |
| 39 | PonudbaPostavkaStoritev | Postavke PonudbaDobavnica, kjer so za vsako storitev opredeljena količina, cena, popust, skupni zneski in davčna stopnja oz. morebitna osvoboditev davka. |
| 40 | Inventura | Inventura vsebuje podatke o zaporedni številki v letu, datumu izdelave, izdelovalcih inventure (prisotni na inventuri) in identifikatorju inventure. |
| 41 | InventuraPostavka | Postavke Inventure. Vsebuje podatke o artiklu, količini, ki se vodi po aplikaciji, realni prešteti količini, ki jo vpiše uporabnik, veljavni nabavni in prodajni ceni ter finančni razliki (v primeru manjka ali viška). |
| 42 | OknoIzbiraDodatnihOpcij | Zaslon za izbiro dodatnih opcij v oknoBlagajna (izdelava ponudbe, izdelava dobavnice, odpiranje predala za denar, popust za celoten račun, vnos napitnine, kopija računa, storno |

| | | |
|---|---|---|
| | | računa ...) |
| 43 | OknoPrijava | Zaslon za prijavo v sistem, ki omogoča vnos uporabniškega imena in gesla ter preverjanje podatkov. |
| 44 | OknoBlagajna | Glavno okno za delo z blagajno. Omogoča dodajanje izdelkov v tabelo, pridruževanje dodatkov izbranim izdelkom, izbiro dodatnih opcij in izbiro načina plačila. |
| 45 | NarociloGostinstvoManager | Kontrolni razred za upravljanje podatkov naročila v bazi, sestavljanja poizvedb SQL in branje podatkov o naročilih iz baze. |
| 46 | RacunGostinsvoManager | Kontrolni razred za upravljanje podatkov računa v bazi, sestavljanja poizvedb SQL in branje podatkov o naročilih iz baze. |
| 47 | IzpisGostinstvo | Razred, ki se uporablja za razne izpiske na tiskalniku POS. POS tiskalniki so značilni po tem, da imajo neskončen papir in omejeno število črk, ki jih lahko izpišejo v eni vrstici. Zato je potrebno implementiranje algoritmov za formatiranje podatkov v določeni širini in številu stolpcev. Primer dela pomožnih razredov za izpis na tiskalnik POS je opisan v poglavju 3.4.1.1 Komponentni diagram trenutnega sistema (Slika 25 in Slika 26). |
| 48 | PonudbaGostManager | Kontrolni razred za upravljanje podatkov ponudbe in dobavnice v bazi, sestavljanja poizvedb SQL in branje podatkov o naročilih iz baze. |
| 49 | OknoIzbiraNacinaPlacila | Zaslon, na katerem je možno izbrati eno od dostopnih načinov plačila (Gotovina, TRR, Različne kreditne kartice, Čeki, Študentski boni – subvencija …). |
| 50 | OknoVecNacinovPlacila | Če želi kupec plačati račun na več različnih načinov, potem zaslonska maska omogoča vnos zneska plačila za posamezen način plačila. |
| 51 | OknoIzbiraMize | Zaslonska maska omogoča grafično izbiro mize, ki so razporejene v skupine (zavihke). Identifikator mize se uporablja pri naročilih. |
| 52 | FiskalSLO | Kontrolni pomožni razred, ki se uporablja za komunikacijo z davčno upravo (prijava poslovnega prostora, pošiljanje računov). |
| 53 | OknoRacuni | Zaslonska maska omogoča pregled (iskanje) že izdanih računov, storno računov in ponovni izpis računov. |
| 54 | OknoZaposleni | Zaslonska maska, ki omogoča dodajanje, urejanje in brisanje podatkov o zaposlenih. |
| 55 | OknoPrejemi | Zaslonska maska omogoča dodajanje, urejanje in storno prejemov. |

Tabela 1: Seznam razredov in kratki opisi.

## *3.3 Diagrami zaporedja*

Z diagrami zaporedja so prikazani dinamičen vidik aplikacije, interakcija in pošiljanje podatkov med komponentami sistema.

V spodaj navedenih diagramih in pri njihovih opisih so podani najpomembnejši koncepti in delovanje aplikacije.

### 3.3.1 Diagram zaporedja izdelave naročila

Na sliki 20 je prikazan postopek izdelave naročila. Natakar se najprej prijavi v aplikacijo, nato vnese želene artikle v tabelo na glavnem oknu. Če je kupec zahteval dodatek oz. je imel opombo, se le-to vpiše in pridruži izdelku. Vpiše se tudi ustrezno količino vsakega izdelka. Pri vsakem vnosu sistem preveri dostopnost izdelkov, tako da s pomočjo normativov, ki povezujejo sestavine in izdelke, izračuna eventualno porabo sestavin. Če določene sestavine ni na zalogi oz. bi z izdelavo trenutnega naročila zaloga prišla v negativno stanje, bo aplikacija opozorila natakarja. Če je med dodajanjem izdelkov prišlo do pomote, se neželeni izdelek izbriše iz tabele.

Ko so vsi izdelki dodani v tabelo, se zaključevanje naročila potrdi z izbiro mize, na katero se naročilo nanaša.

Po izbiri mize se naročilo shrani v bazo.

Če je bil vnos v bazo uspešen, se nadaljuje izpis posameznih delov naročila na določene tiskalnike POS in signal se pošlje do sistema za opozarjanje, da je bilo novo naročilo vpisano.

Če na naročilu obstaja pijača, se vsi izdelki, ki spadajo v to skupino, izpišejo na tiskalnik pri točilnem pultu.

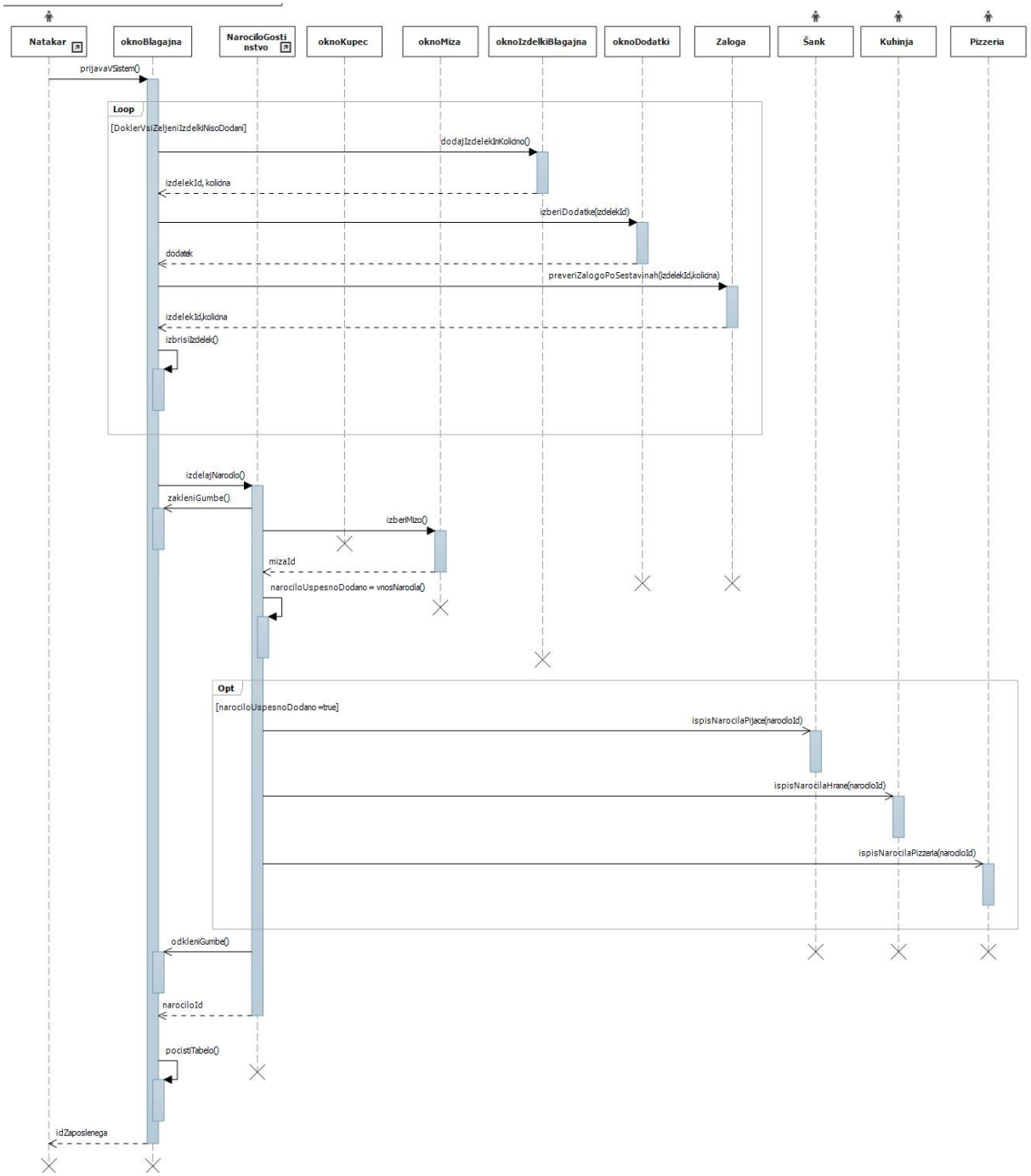
Če na naročilu obstaja hrana, se vsi izdelki, ki spadajo v to skupino, izpišejo na tiskalnik v kuhinji.

Če na naročilu obstaja pica, se vsi izdelki, ki spadajo v to skupino, izpišejo na tiskalnik v tretjem prostoru (piceriji).

Na koncu se glavno okno počisti in postavi v stanje pričakovanja novega naročila.

Če je bila transakcija, vpisa v bazo, neuspešna, program počaka določen čas in skuša narediti vnos v bazo še dvakrat. Če tudi po tretjem poskusu ne uspe vnos v bazo, se uporabniku sporoči, da je prišlo do napake in naj poskusi ponovno.

Diagram zaradi zapletenosti ne prikazuje alternativnih dogodkov in izjem.




Slika 20: Diagram zaporedja izdelave naročila.

### 3.3.2 Diagram zaporedja izdelave računa

Račun lahko izdelamo na več načinov. Slika 21 prikazuje direktno izdelavo računa brez predhodno izdelanega naročila, ponudbe ali dobavnice.

Ko se uporabnik prijavi v aplikacijo, pride do glavnega okna blagajne.

Potem vnese želene artikle v tabelo na glavnem oknu. Vpiše tudi ustrezno količino vsakega izdelka. Pri vsakem vnosu sistem preveri dostopnost izdelkov, tako da s pomočjo normativov, ki povezujejo sestavine in izdelke, izračuna eventualno porabo sestavin. Če določene sestavine ni na zalogi oz. bi z izdelavo trenutnega računa zaloga prišla v negativno stanje, bo aplikacija opozorila natakarja. Če je med dodajanjem izdelkov prišlo do pomote, se neželeni izdelek izbriše iz tabele.

Ko so vsi izdelki dodani v tabelo, se zaključevanje potrdi z izbiro kupca. Če se korak izbire kupca preskoči, potem se račun poveže s predpostavljenim maloprodajnim kupcem.

Nato sledi izbira načina plačila. Če je izbran način plačila „Več načinov plačila“, se odpre okno, kjer se lahko pri vsakem načinu plačila vpiše znesek. Vsota vseh zneskov mora biti enaka znesku računa.

Aplikacija v nadaljevanju zaklene gumbe, da se vnos računa slučajno ne bi podvojil.

Račun se vpiše v bazo. V isti transakciji je treba spremeniti povezane podatke (spremeniti zalogo artiklov, tj. sestavin, spremeniti kartico artikla in posodobiti trgovsko knjigo, če je izdelek v skupini trgovskih izdelkov).

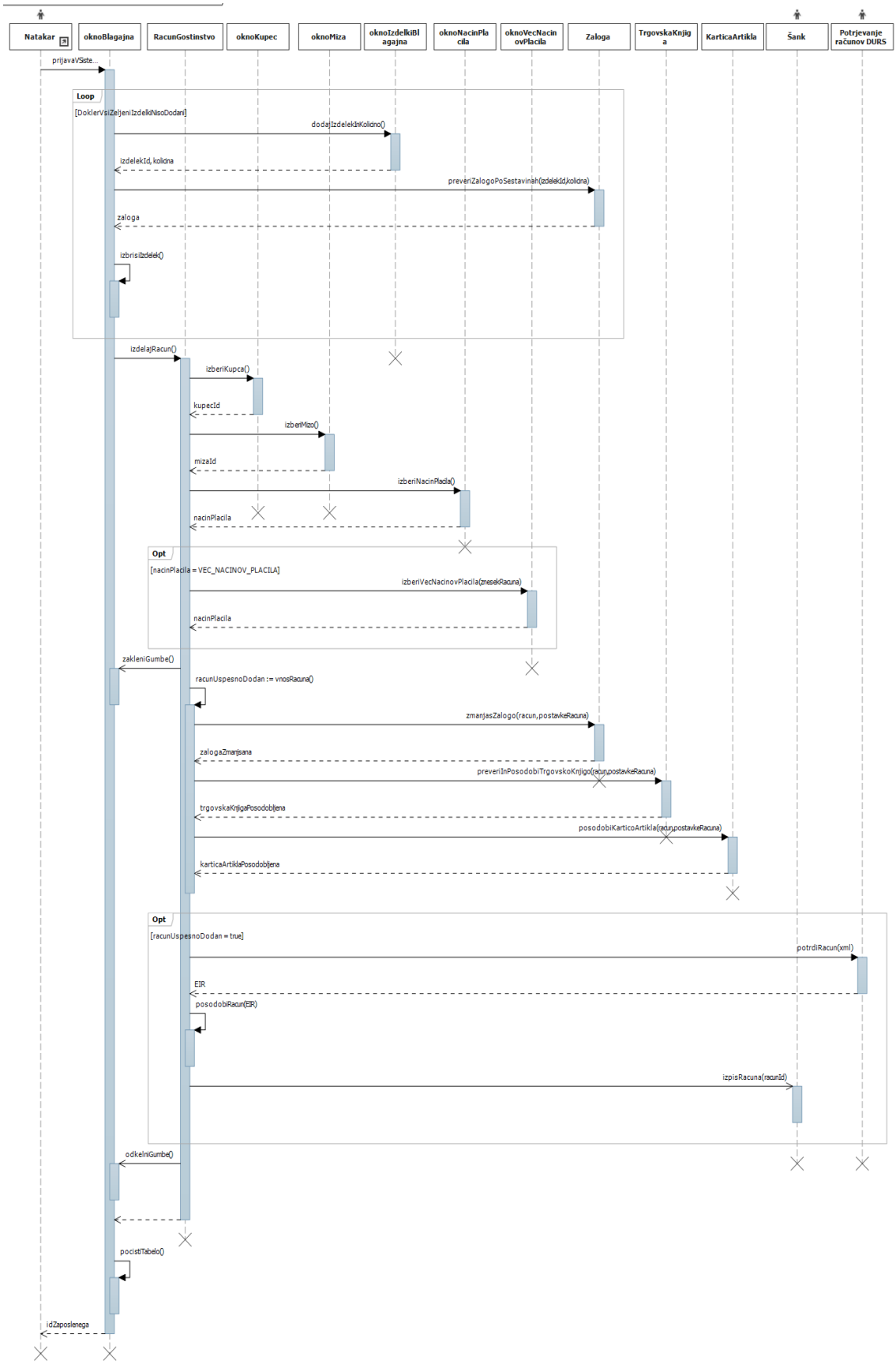
Če se račun uspešno vpiše v bazo, se nadaljuje s procesom potrjevanja računa prek servisov DURS-a.

Aplikacija počaka določen čas na odgovor. Če ga ne dobi v pričakovanem času, bo za naknadno potrjevanje poskrbel modul za avtomatsko obdelavo podatkov.

Če odgovor vsebuje podatek EOR, bo sistem posodobil podatek o potrjevanju v podatkovni bazi.

Na koncu se račun izpiše na tiskalnik POS (ali navadni) – odvisno od tipa kupca in nastavitev parametrov aplikacije.

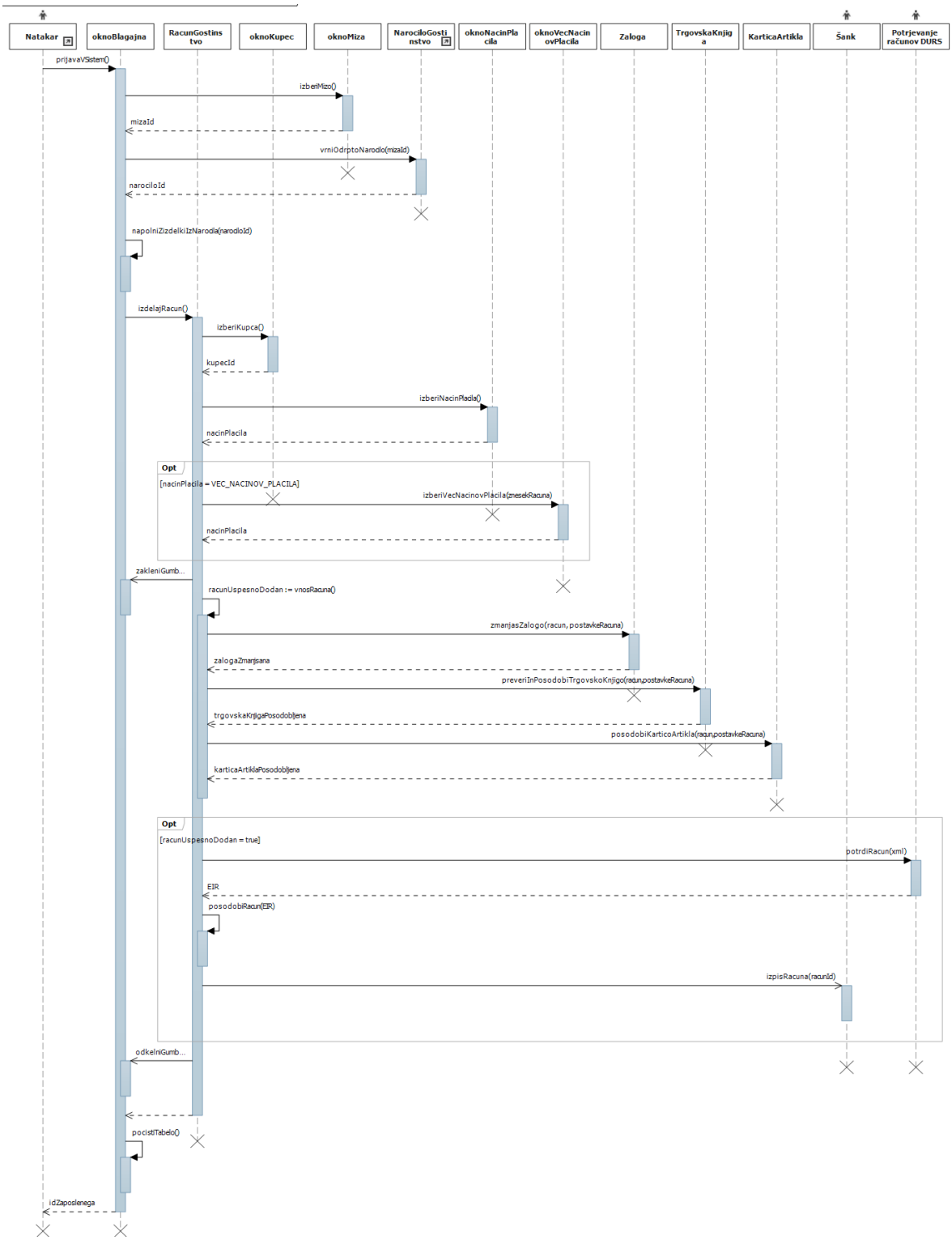
Po končanem procesu (uspešno ali neuspešno) se počisti okno blagajne in gumbi se odklenejo.




Slika 21: Diagram zaporedja izdelave računa.

### 3.3.3 Diagram zaporedja izdelave računa iz naročila

Izdelava računa iz naročila se začne z izbiro mize, ki mora imeti vsaj eno naročilo odprto. S pomočjo identifikacijske številke mize pridobimo podatke o postreženih izdelkih in količinah, s čimer napolnimo tabelo izdelkov. Od tukaj naprej je proces izdelave enak procesu izdelave računa – direktno, opisanega v prejšnjem primeru.




Slika 22: Diagram zaporedja izdelave računa iz naročila.

### 3.3.4 Diagram zaporedja izdelave vhodno-izhodnih dokumentov

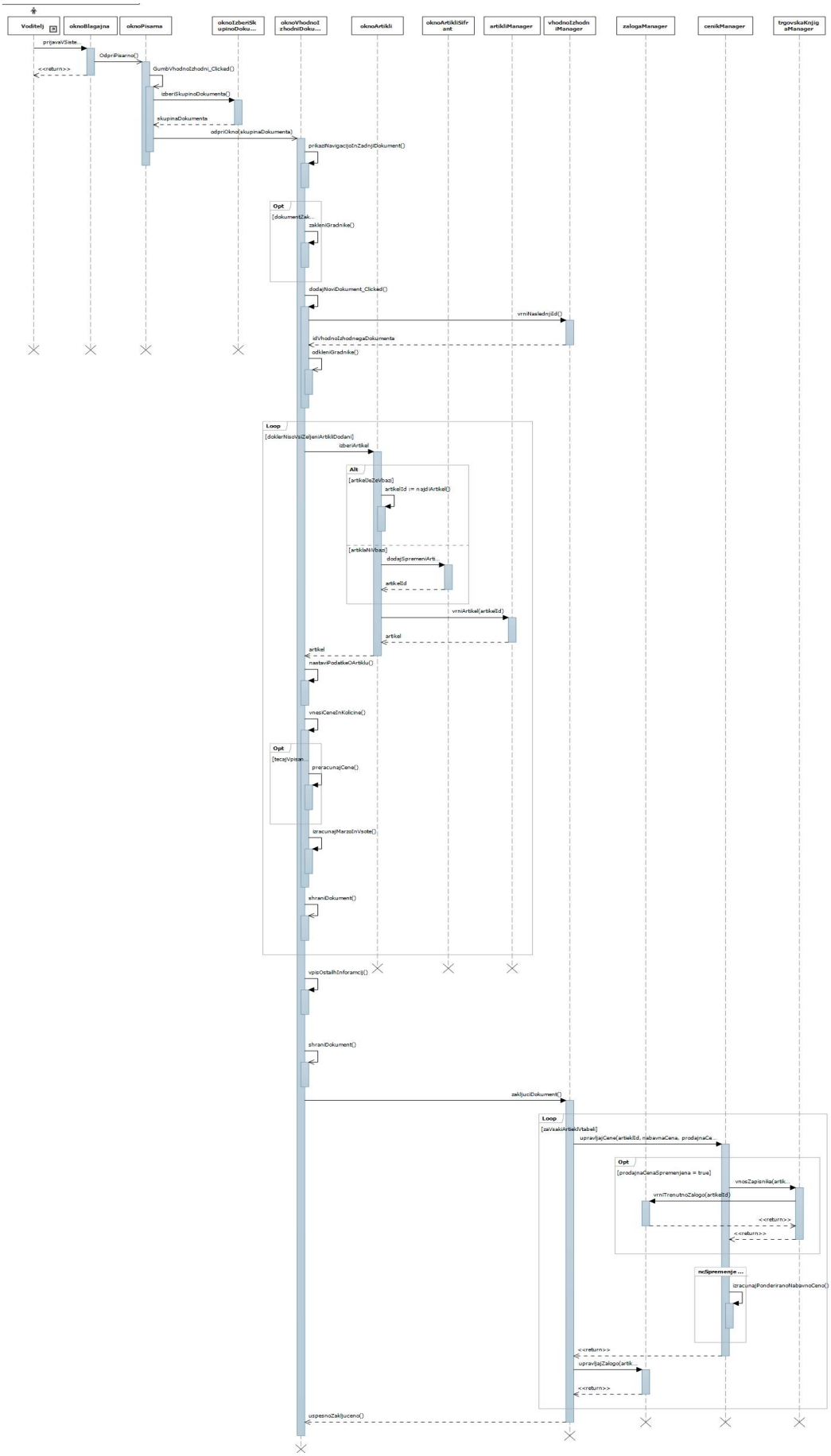
Na spodnji sliki je prikazan vnos vhodno-izhodnih dokumentov. To so prejemi, medskladiščnice, odpisi ...

Vnos je omogočen voditelju oz. uporabniku, ki ima dostop do pisarne. To je t. i. *backoffice* del aplikacije.

Po vstopu v pisarno se s klikom na gumb „Prejemi“ odpre okno z izbiro skupine dokumenta. Po izbiri skupine dokumenta se odpre okno z izbranimi dokumenti. V oknu se vedno prikaže zadnji vpisani dokument (če le-ta obstaja). Sproži se postopek nastavljanja gradnikov okna v odvisnosti, ali je dokument že zaključen ali je dokument še v fazi izdelave in urejanja. Za vnos novega dokumenta je treba klikniti na gumb „Dodaj novi dokument“. S to akcijo se okno počisti in se pripravi okno za dodajanje artiklov na nov dokument. Doda se želene artikle (s pomočjo iskalnika, črtne kode, šifre ...). Če artikel ni vpisan v bazo, se s pomočjo novega okna za vnos artikla doda nov artikel. Za vsak artikel se poljubno vpiše količino, nabavno in prodajno ceno. Aplikacija bo po vsaki spremembi ustrezno spremenila maržo, vmesne vsote in vsote celotnega dokumenta. Če je artikel že v bazi, se ustrezno nastavi zadnje veljavne cene (ponderirana nabavna cena in prodajna cena). Če gre za izdelavo prejema – izračuna s področja tuje valute (imamo fakturo v tuji valuti), se iz vpisanega tečaja preračuna cene v domicilni valuti.

Dokument lahko poljubnokrat shranimo, zapremo in ponovno odpremo za urejanje, vse dokler ga ne zaključimo. Aplikacija poskrbi tudi za samodejno shranjevanje vsako minuto, tako da ob morebitnem izpadu električnega toka obdrži čim bolj sveže informacije.

Kadar je dokument pripravljen za zaključevanje, uporabnik zažene akcijo zaključevanja. S tem postopkom se sproži naslednji proces: za vsak artikel na dokumentu se preveri, če je prodajna cena spremenjena glede na zadnjo ceno. Če je spremenjena in je trenutna zaloga večja od 0, je treba sestaviti zapisnik o spremembi cen za artikle, ki so trenutno na zalogi. Za vsak artikel je treba preračunati tudi ponderirano nabavno ceno (če je trenutno na zalogi). Na koncu je treba urediti oz. spremeniti še zalogo artikla. Če se celoten proces, ki se odvija v eni transakciji, zaključi uspešno, se posodobi podatke o dokumentu, in sicer informacija, da je dokument zaključen.

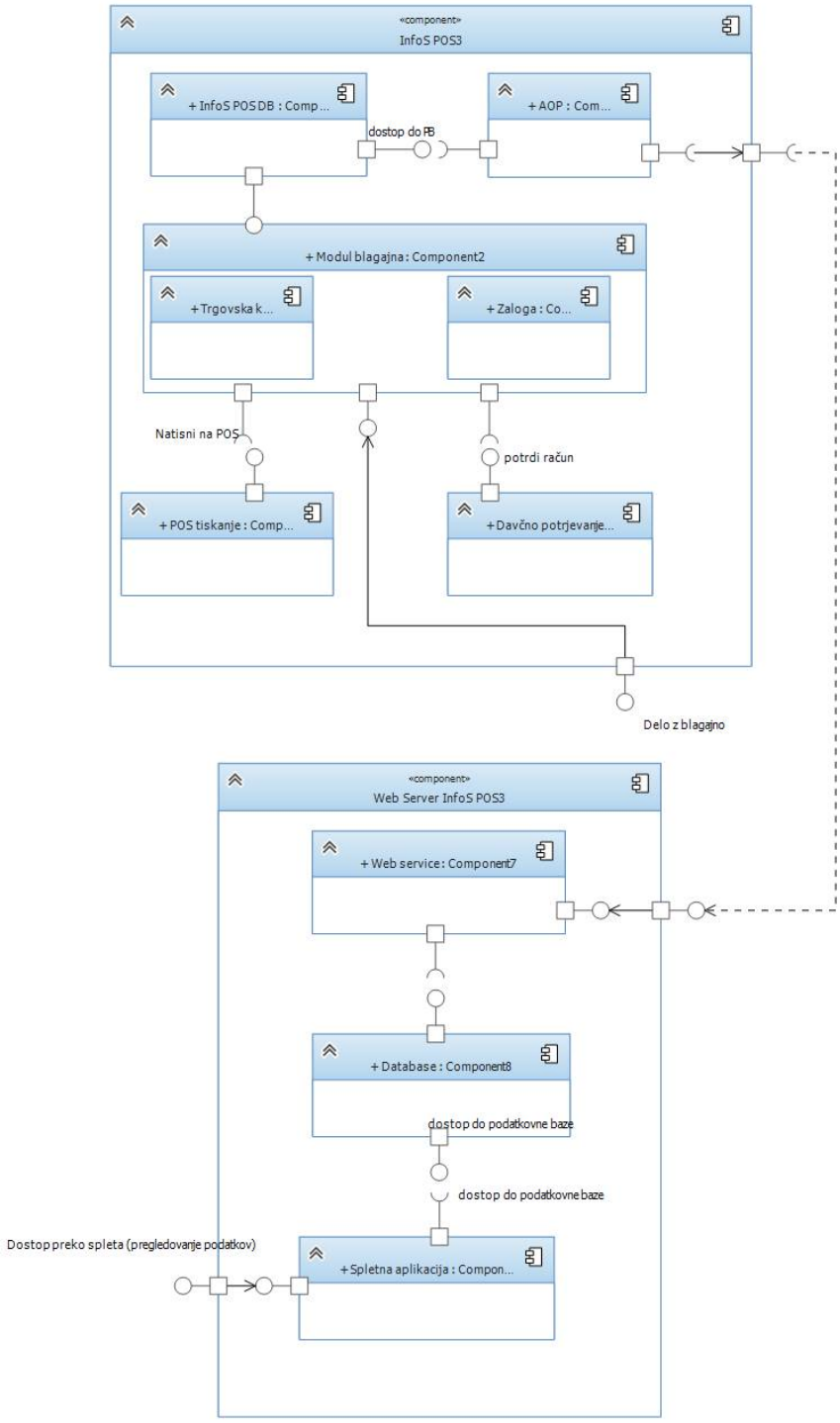



Slika 23: Diagram zaporedja vhodno-izhodnih dokumentov.

## 3.4 Komponentni diagram

### 3.4.1.1 Komponentni diagram trenutnega sistema

Diagram na spodnji sliki prikazuje, kako je sestavljena aplikacija in katere dele aplikacije se lahko spremeni/posodobi/zamenja, ne da bi zamenjali celoten sistem.




Slika 24: Komponentni diagram trenutnega sistema.

| Komponenta | Podkomponenta | Opis |
|---|---|---|
| InfoS POS | InfoS_POS_DB | Podatkovna baza, v kateri so shranjeni vsi podatki v enem poslovnem prostoru. V podatkovno bazo se lahko zapišejo podatki od več odjemalcev (če ima recimo poslovni prostor več blagajn in pisarniški del na ločenem računalniku). Podatkovna baza je lahko nameščena na zasebnem strežniku ali na istem računalniku, na katerem je nameščena tudi programska oprema oz. aplikacija InfoS POS. Če je nameščena na zasebnem strežniku, mora biti povezana z odjemalci (aplikacijami) prek ustreznega omrežja (po navadi LAN ali WLAN). |
| InfoS POS | AOP – avtomatska obdelava podatkov | Sistem za avtomatsko obdelavo podatkov (3.1.4 ) skrbi za potrjevanje nepotrjenih računov, sinhronizacijo na spletni stražnik. Je popolnoma avtomatizirana komponenta, katere akcije so vezane na časovne alarme. Skrbi, da se vsi podatki, ki so pripravljeni na sinhronizacijo, prenesejo na spletni strežnik, ne da bi za to skrbel uporabnik. Na enak način skrbijo tudi za naknadno potrjevanje nepotrjenih računov. Do nepotrjenih računov lahko pride iz več različnih razlogov (izpad omrežja oz. povezave do interneta, preobremenitev strežnikov za potrjevanje ...). Ko se težave odpravijo, sistem avtomatsko poskrbi za davčno potrjevanje in vpis rezultatov v bazo. |
| InfoS POS | Blagajna | Osrednja komponenta, ki nudi vse funkcionalnosti, in uporabniški vmesnik. Je glavna komponenta, prek katere uporabnik izdeluje naročila, ponudbe, dobavnice, izdaja račune, vzdržuje šifrante, pregleduje poročila, izdeluje izračune in vhodno-izhodne dokumente (prejeme, medskladiščnice, odpise, povračila ...). Vsebuje tudi dve podkomponenti, ki sta zadolženi za vzdrževanje trgovske knjige in zaloge. Zaloga se spreminja z zaključevanjem posameznega vhodno-izhodnega dokumenta, z izdelavo računa in z izdelavo dobavnice. Trgovska knjiga se spreminja z zaključkom vhodno-izhodnih dokumentov, ki vsebuje trgovske artikle, z zapisnikom o spremembi cen, ki so posledica neposredne spremembe cen za artikle, katerih trenutna zaloga je različna od 0, ali kot posledica popusta na računu pri prodaji izdelkov. |
| InfoS POS | Blagajna - trgovska knjiga | Komponenta, v kateri se vzdržuje podatke o trgovski knjigi, skrbi za vnos, spremembe, brisanje zapisov v podatkovni bazi, za pregled in izpisovanje trgovske knjige. Je podkomponenta glavne komponente Blagajne. |
| InfoS POS | Zaloga | Komponenta, ki skrbi za zalogo artiklov. Prek vmesnikov lahko spreminjamo stanje zaloge artiklov (dodajanje, brisanje, spremembe) v določenem skladišču. Nudi tudi vmesnike za pregled in izpis zaloge za posamezno skladišče ali za vsa skladišča. Je podkomponenta glavne komponente Blagajne. |
| InfoS POS | POS tiskanje | Komponenta, ki skrbi za tisk računov na tiskalnike POS. V njej je vgrajena logika za tisk na tiskalnike različnih zmogljivosti. Tiskalniki POS so značilni po tem, da lahko v eni vrstici natisnejo omejeno število znakov v točno določeni (največkrat vgrajeni v tiskalnik) pisavi. Pošiljamo jim posebne kode za poravnavo pisave in ostalo |

| | | |
|---|---|---|
| | | formatiranje (krepko pisavo, velikost pisave), odpiranje predala, signal za zvonec ... Nekaj največjih svetovnih proizvajalcev ima določene lastne kode in komande za prej opisane funkcionalnosti, zato je treba kode prilagoditi, odvisno od znamke in modela tiskalnika. |
| Spletni strežnik | | Namenjen je shrambi in pregledovanju podatkov iz lokalnih blagajn. Sestavljen je iz treh podkomponent. Prva nudi vmesnik za vnos podatkov, druga je podatkovna baza, tretja pa spletno mesto za pregledovanje podatkov. |
| Spletni strežnik | Spletni servis | Spletni servis nudi vmesnik za sinhronizacijo podatkov iz lokalne blagajne na spletno podatkovno bazo v oblaku – komunicira s komponento AOP, ki pošilja podatke iz lokalnih odjemalcev in lokalnih podatkovnih baz. |
| Spletni strežnik | Spletna pod. baza | Spletno podatkovno bazo uporabljata spletni servis za sinhronizacijo in spletna aplikacija za prikaz podatkov. Vsebuje podmnožico podatkov (sinhronizirajo se samo podatki o prodaji, ne pa tudi vhodno-izhodni dokumenti) lokalne podatkovne baze na odjemalcu z določenim časovnim zamikom. Posredno pa ima funkcijo tudi digitalnega arhiva podatkov – pri morebitni izgubi podatkov na lokalni blagajni se le-ti lahko rekonstruirajo iz spletne shrambe. |
| Spletni strežnik | Spletna aplikacija | Spletna aplikacija nudi pregled izdanih računov iz posameznih lokacij in ostale preglede (po artiklih, davčnih skupinah, skupinah artiklov, kupcih, zaposlenih, obdobju, urah ...). Tukaj lahko voditelj, upravljalec ali lastnik pregleduje podatke za posamezno prodajno mesto in tudi agregirane podatke za vsa prodajna mesta skupaj. Aplikacija nudi tudi medsebojno primerjavo različnih prodajnih mest znotraj ene davčne številke. Omogoča pregled prodaje in ostala poročila s katerekoli lokacije, tudi ko so lokalne blagajne izklopljene. |

Tabela 2: Seznam in opis komponent.

Neposredno tiskanje na tiskalnik POS dela največ težav neizkušenim razvijalcem, zato smo v nadaljevanju navedli kodo, ki se uporablja v teh primerih (Slika 25 in Slika 26).

```
public class CommandHelper
{
    /// <summary>Komanda za odpiranje predala za denar.</summary>
    /// <param name="codes">Kode za odprianje predala (odvisno od tiskalnika). Če je ni se nastavi na privzeto vrednost.</param>
    /// <returns>Vrne komando za odpiranje predala.</returns>
    2 references
    public static String OpenCashDrawer(int[] codes = null)
    {
        if (codes == null) { codes = new int[] { 27, 112, 0, 25, 250 }; }
        StringBuilder sequence = new StringBuilder();
        foreach (int code in codes)
            sequence.Append((char)code);

        return sequence.ToString();
    }

    /// <summary>Komanda za rezanje papirja pri izpisu računa ali naročila.</summary>
    /// <param name="codes">Kode za rezanje (odvisno od tiskalnika). Če je ni se nastavi na privzeto vrednost.</param>
    /// <returns>Vrne komando za rezanje papirja.</returns>
    1 reference
    public static String CutPaper(int[] codes = null)
    {
        if (codes == null) {codes = new int[] { 27, 105, 0, 25 }; }
        StringBuilder sequence = new StringBuilder();
        foreach (int code in codes)
            sequence.Append((char)code);

        return sequence.ToString();
    }
    /// <summary>Inicijalizacija tiskalnika. Na začetku je treba tiskalnik nastaviti in določiti kodno stran.</summary>
    /// <param name="codes">Kode za točno določeni POS tiskalnik. Če je ni se nastavi na privzeto vrednost.</param>
    /// <returns>Vrne komando za nastavitev tiskalnika.</returns>
    2 references
    public static String InitializePrinter(int[] codes = null)
    {
        if (codes == null) { codes = new int[] { 27, 64 }; }
        StringBuilder sequence = new StringBuilder();
        foreach (int code in codes)
            sequence.Append((char)code);

        return sequence.ToString();
    }
    /// <summary>Na začetku je treba tiskalnik nastaviti in določiti kodno stran.</summary>
    /// <param name="codes">Šifra kodne strani (odvisno od tiskalnika), če je ni se nastavi na privzeto vrednost.</param>
    /// <returns>Vrne komando za nastavitev kodne strani</returns>
    1 reference
    public static String SetCharacterSet(int[] codes = null)
    {
        if (codes == null) {codes = new int[] { 27, 116, 18 }; }
        StringBuilder sequence = new StringBuilder();
        foreach (int code in codes)
            sequence.Append((char)code);

        return sequence.ToString();
    }
    /// <summary> Tiskalnik bo nadaljeval tiskanje z sredinsko poravnavo</summary>
    0 references
    public static String PrintCenterText()...
    /// <summary>Tiskalnik bo nadaljeval tiskanje z levo poravnavo </summary>
    0 references
    public static String PrintLeftText()...
    /// <summary>Komanda za izpis praznih vrstic.</summary>
    /// <param name="pNumLines">Število praznih vrstic.</param>
    /// <param name="codes">Kode za prazne vrstice (odvisno od tiskalnika), če je ni se nastavi na privzeto vrednost.</param>
    /// <returns>Vrne komando za tisk praznih vrstic.</returns>
    1 reference
    public static String FeedPaper(int pNumLines, int[] codes = null)
    {
        if (codes == null) { codes = new int[] { 27, 100 }; }
        StringBuilder sequence = new StringBuilder();
        foreach (int code in codes)
            sequence.Append((char)code);

        sequence.Append((char)pNumLines);
        return sequence.ToString();
    }
}
```

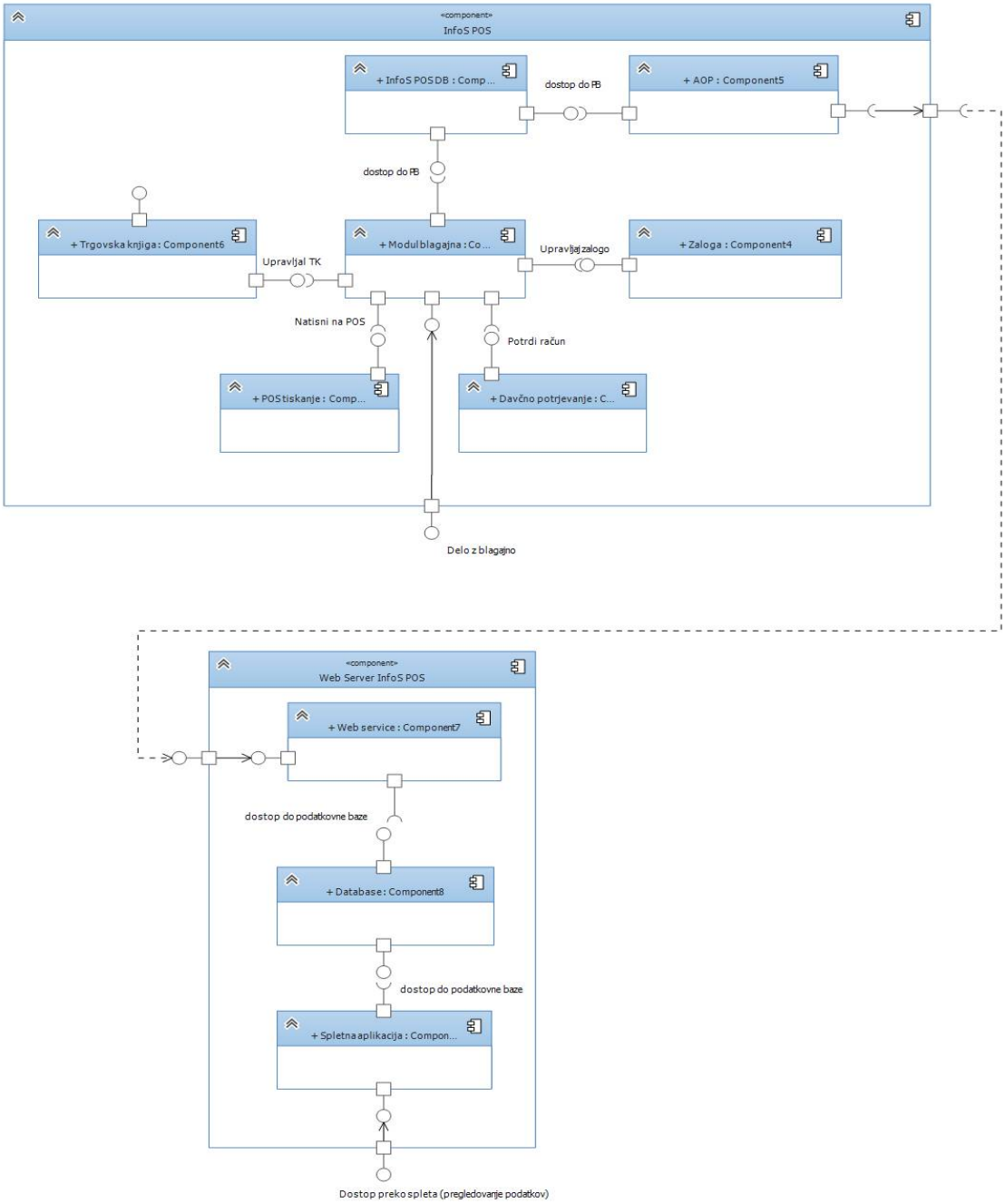
Slika 25: Izpis na tiskalnik POS – primer kode v programskem jeziku C# – prvi del.

```
/// <summary>
/// Pošilj podatke v tiskalnik.
/// </summary>
/// <param name="szPrinterName">Naziv tiskalnika</param>
/// <param name="pBytes">Podatki v obliki byte</param>
/// <param name="dwCount">Število znakov</param>
/// <returns>true - uspeh, false - neuspeh</returns>
2 references
public static bool SendBytesToPrinter(string szPrinterName, IntPtr pBytes, Int32 dwCount)
{
    Int32 dwError = 0, dwWritten = 0;
    IntPtr hPrinter = new IntPtr(0);
    DOCINFOA di = new DOCINFOA();
    bool bSuccess = false; // Predpostavi neuspeh vse dokler ne pridemo do uspešnega tiskanja.

    di.pDocName = "POSprinting";
    di.pOutputFile = null;
    di.pDataType = "RAW";

    // Odpri tiskalnik.
    if (OpenPrinter(szPrinterName.Normalize(), out hPrinter, IntPtr.Zero))
    {
        // Zaženi dokument..
        if (StartDocPrinter(hPrinter, 1, di))
        {
            // Zaženi stran.
            if (StartPagePrinter(hPrinter))
            {
                // Piši podatke.
                bSuccess = WritePrinter(hPrinter, pBytes, dwCount, out dwWritten);
                EndPagePrinter(hPrinter);
            }
            EndDocPrinter(hPrinter);
        }
        ClosePrinter(hPrinter);
    }
    // Če je prišlo do napake poskusi pridobiti informacije o napaki.
    if (bSuccess == false)
    {
        dwError = Marshal.GetLastWin32Error();
    }
    return bSuccess;
}
0 references
public static bool SendFileToPrinter(string szPrinterName, string szFileName)
{
    // Odpri datoteko.
    FileStream fs = new FileStream(szFileName, FileMode.Open);
    // Binarno preberi datoteko.
    BinaryReader br = new BinaryReader(fs);
    Byte[] bytes = new Byte[fs.Length];
    bool bSuccess = false;
    // Naredi kazalnik.
    IntPtr pUnmanagedBytes = new IntPtr(0);
    int nLength;

    nLength = Convert.ToInt32(fs.Length);
    // Preberi podatke v tabelo.
    bytes = br.ReadBytes(nLength);
    // Alociraj pomnilnik.
    pUnmanagedBytes = Marshal.AllocCoTaskMem(nLength);
    // Kopiraj v neupravljano tabelo.
    Marshal.Copy(bytes, 0, pUnmanagedBytes, nLength);
    // Pošalji podatke na tiskanje.
    bSuccess = SendBytesToPrinter(szPrinterName, pUnmanagedBytes, nLength);
    // Osvobodi pomnilnik.
    Marshal.FreeCoTaskMem(pUnmanagedBytes);
    return bSuccess;
}
4 references
public static bool SendStringToPrinter(string szPrinterName, string szString)
{
    IntPtr pBytes;
    Int32 dwCount;
    // Število znakov v nizu.
    dwCount = szString.Length;
    // Pretvori besedilo v unicode.
    pBytes = Marshal.StringToCoTaskMemAnsi(szString);

    // Pošlji podatke na tiskanje.
    SendBytesToPrinter(szPrinterName, pBytes, dwCount);
    Marshal.FreeCoTaskMem(pBytes);
    return true;
}
```

Slika 26: Izpis na tiskalnik POS – primer kode v programskem jeziku C# – drugi del.

### 3.4.1.2 Komponentni diagram možne izboljšave

Na spodnji sliki je prikazana možna izboljšava sistema z metodo dekompozicije na več komponent, ki bi jih lahko vzdrževali in zamenjali neodvisno od ostalih delujočih komponent. Izboljšava bi vplivala tudi na avtomatske posodobitve, ki potekajo prek tehnologije MS Click Once (aplikacija pri zagonu vedno preveri, če obstaja nova verzija aplikacije na strežniku za posodobitev, in če obstaja, se aplikacija avtomatsko nadgradi. Click Once ima vgrajeno logiko nadomestitve samo tistih datotek, ki so bile spremenjene).




Slika 27: Komponentni diagram možne izboljšave sistema.

## 3.5 Diagram postavitve

S spodnje slike 28 diagrama postavitve lahko razberemo različne načine postavitve celotnega sistema.

V prvem modelu je aplikacija nameščena na računalnik skupaj s podatkovno bazo. Na spletnem strežniku je postavljen aplikacijski strežnik, na katerem se izvajajo spletni servisi za sinhronizacijo podatkov. Ker imajo po navadi vse komercialne dostopne rešitve domovanja podatkovne baze na ločenih strežnikih od aplikacijskih strežnikov, je tudi v tej rešitvi izbran omenjeni model. Aplikacijski strežnik na spletu poganja tudi spletno aplikacijo za pregled poslanih podatkov. Do aplikacije se dostopa prek navadnega spletnega brskalnika z uporabo računalnika, prenosnika, tablice ali drugih podobnih naprav.

Drugi model prikazuje postavitev v bolj zapletenih poslovnih objektih, kjer je zaradi konfiguracije prostorov v objektu in velikosti samega objekta nujna uporaba več kot ene blagajne. V tem primeru lahko podatkovno bazo in sistem s podatkovno bazo postavimo na interni strežnik. Na vsako blagajno je nameščena aplikacija, ki komunicira s podatkovno bazo prek lokalnega omrežja. Podatki se sinhronizirajo na vsaki blagajni posebej na spletni servis. Ostali del sistema je enak zgoraj opisanemu prvemu modelu.

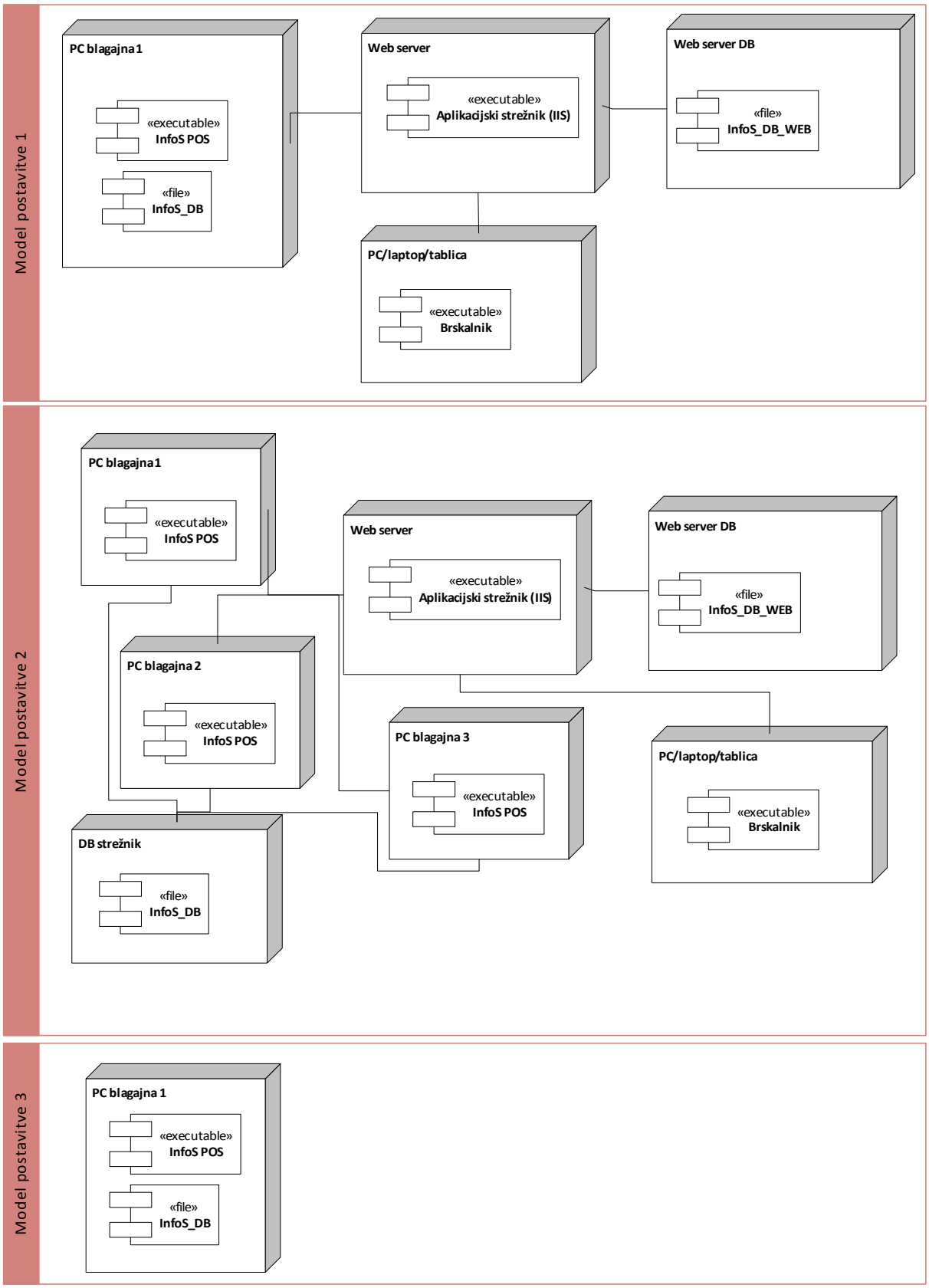
Tretji model je namenjen najbolj enostavnim sistemom, kjer ni potrebe po spletnem pregledovanju, v objektu pa zadošča namestitev ene blagajne. V tem primeru na računalnik namestimo aplikacijo in podatkovno bazo. Internetno povezavo potrebujemo v tem primeru samo zaradi potrjevanja računov.

Možnih je seveda še več različnih načinov postavitve in kombinacij treh opisanih modelov, vendar zaradi velikega števila različnih kombinacij ni smiselno opisati vseh.

Eden od takšnih primerov je tudi, kadar ima podjetje več prodajnih mest/poslovnih prostorov na različnih geolokacijah, v tem primeru so lahko možne kombinacije blagajn na določeni lokaciji, na kateri je nameščena podatkovna baza, kot tudi aplikacije, vse pa sinhronizirajo podatke na isti spletni strežnik prek povezave WAN.

Druga možnost je, kadar je v enem poslovnem prostoru več blagajn in je podatkovni strežnik ločen od aplikacije oz. blagajn. Na drugih lokacijah je pa samo po ena blagajna v poslovnem prostoru in na istem računalniku je nameščena tudi podatkovna baza. Vsi skupaj pa zopet pošiljajo podatke na isti spletni strežnik, ne da bi se zavedali eden drugega.

**Modeli postavitve:**

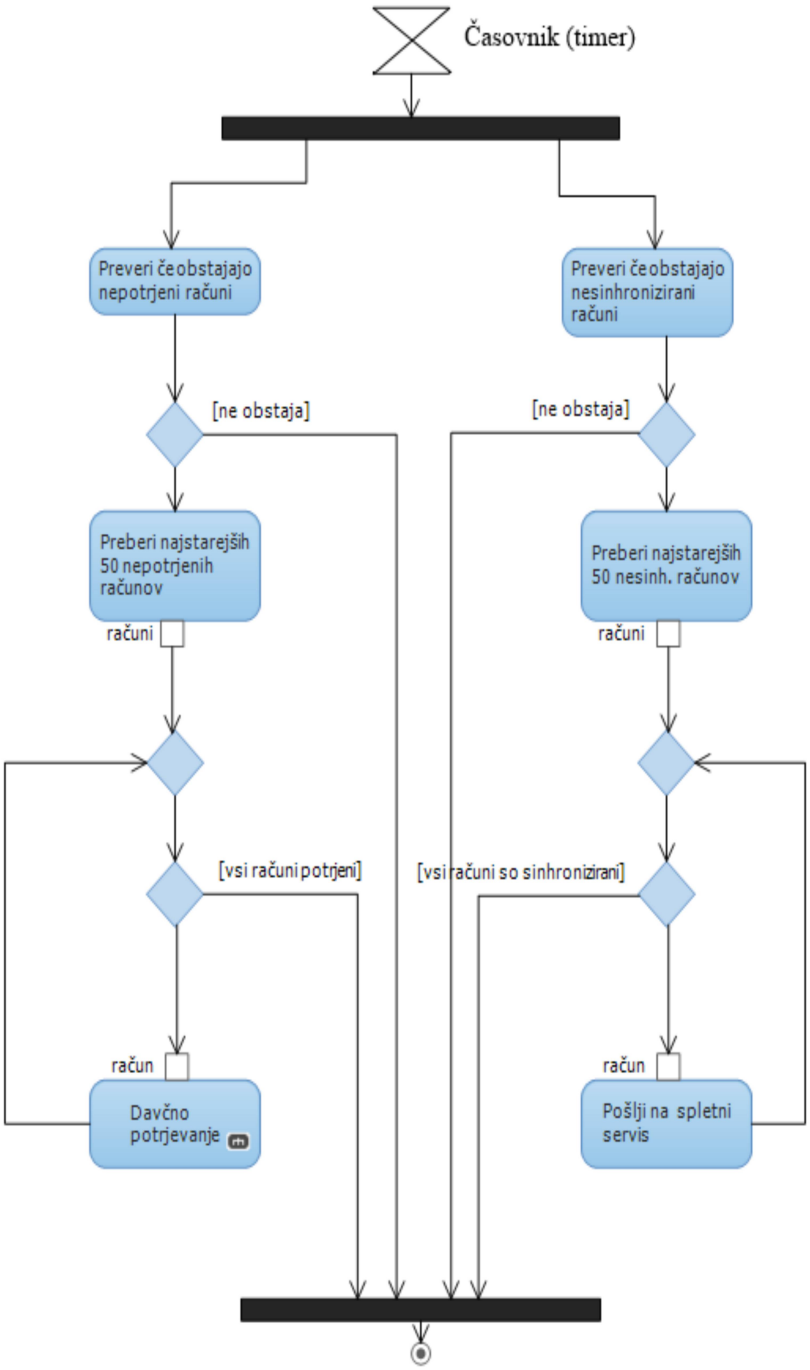



Slika 28: Diagram možnih postavitev sistema.

## 3.6 Diagrami aktivnosti

Z diagrami aktivnosti ponazorimo dinamičen pogled sistema. V diagramih so podani najpomembnejši dogodki, vejitve, združitve in vzporedni tokovi dogodkov. So pogled na obnašanje določenega dela sistema z višjega abstraktnega nivoja, kjer se ne spuščamo v podrobnosti tehnične rešitve.

### 3.6.1 Diagram aktivnosti avtomatske obdelave podatkov




Slika 29: Diagram aktivnosti avtomatske obdelave podatkov AOP.

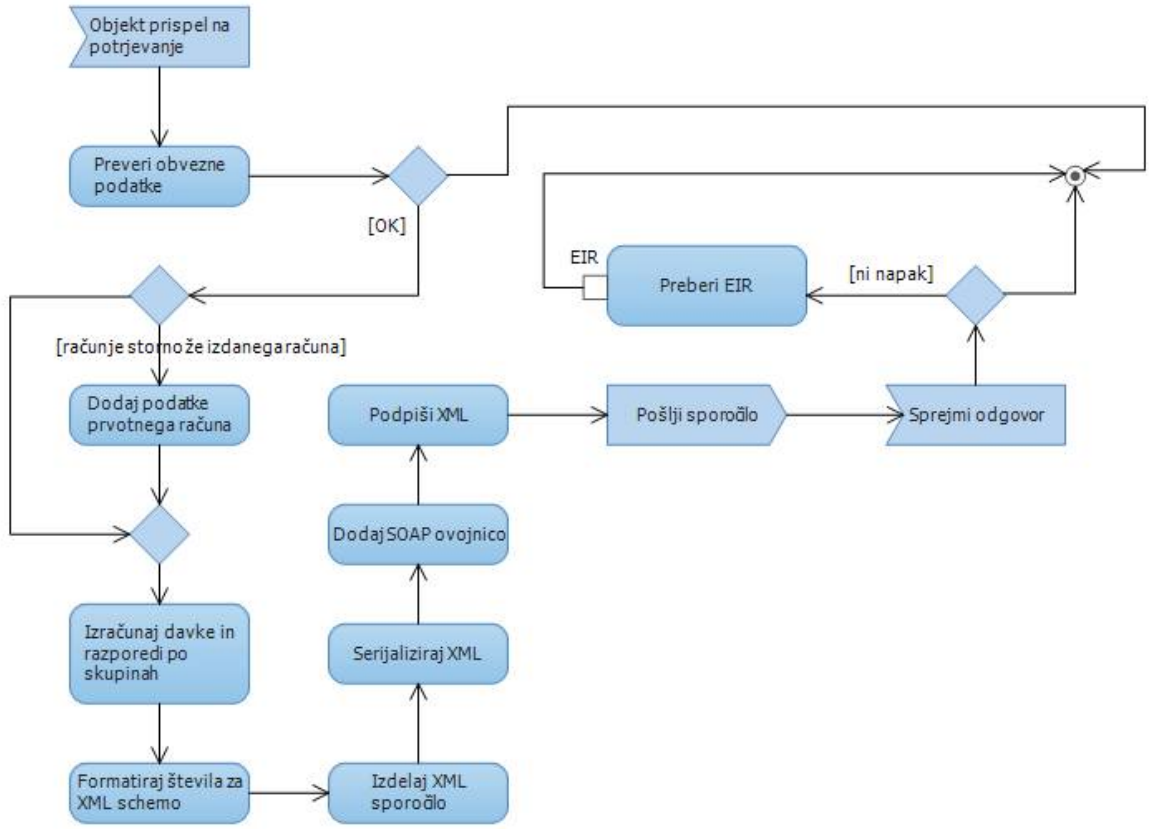
Slika 29 prikazuje ponavljajočo se aktivnost avtomatske obdelave podatkov. Ta se dogaja v dveh ločenih nitih.

Prva nit skrbi za naknadno davčno potrjevanje nepotrjenih računov (ki iz različnih razlogov niso potrjeni pri izdelavi). Najprej se preveri, če obstajajo nepotrjeni računi. Če ne obstaja noben nepotrjen račun, se aktivnost konča. Če obstajajo nepotrjeni računi, se iz podatkovne baze prebere največ 50 takšnih računov in se jih poskuša davčno potrditi. Po uspešnem končanju se računi označijo kot potrjeni, sicer bodo obravnavani v naslednji iteraciji.

Analogija velja tudi za pošiljanje podatkov v oblak oz. sinhronizacijo na spletni strežnik.

### 3.6.2 Diagram aktivnosti davčnega potrjevanja

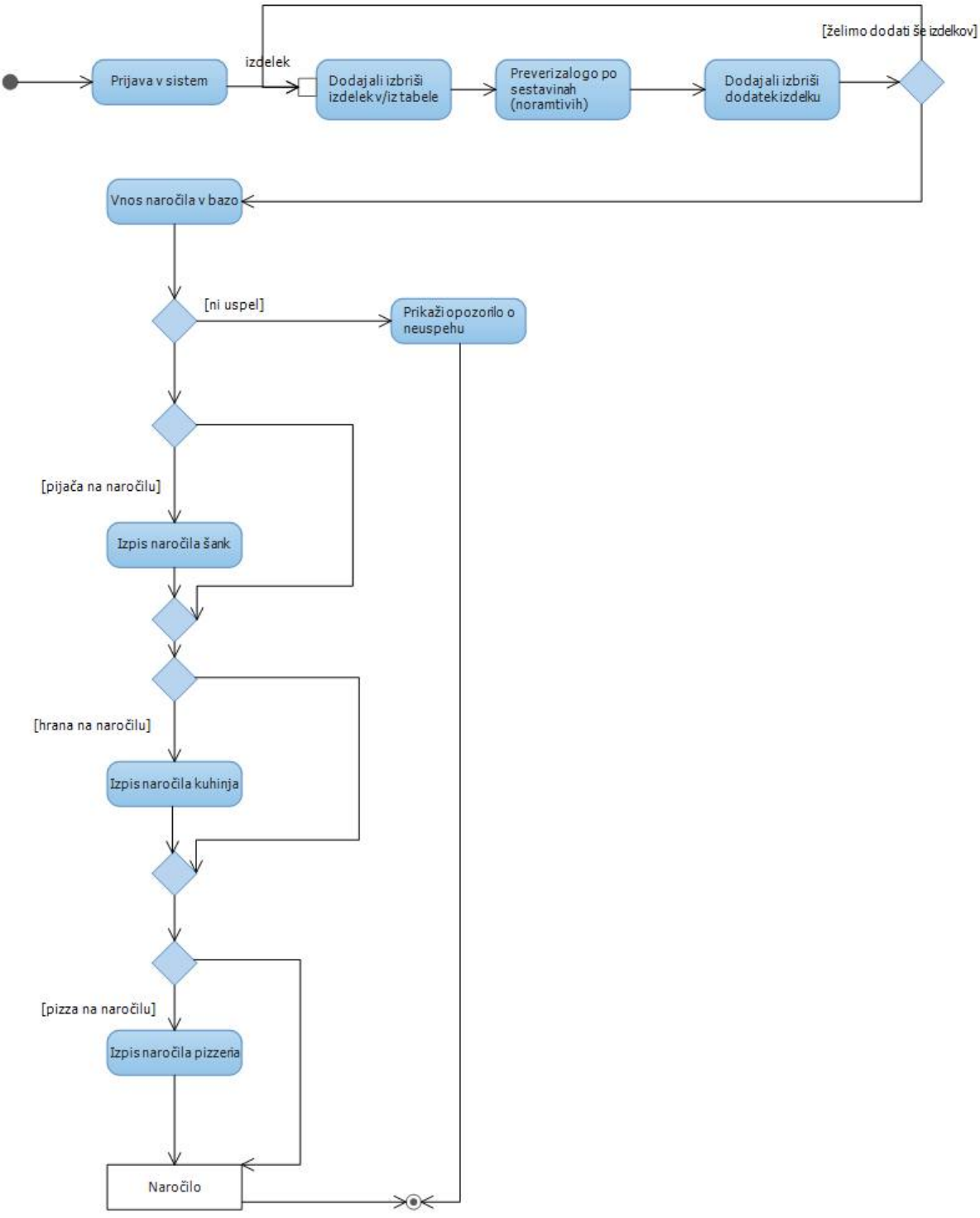
Diagram aktivnosti na sliki 30 opisuje proces davčnega potrjevanja računa. Po klicu metode za davčno potrjevanje se preveri vse obvezne podatke (davčno številko, certifikat za podpisovanje, certifikat za komunikacijo ...). Če so pravilni, preverimo, če je račun storno drugega računa. V tem primeru moramo prebrati iz baze še podatke prvotnega računa. Nato razdelimo davke po davčnih stopnjah in formatiramo števila v skladu s shemo XSD. Nato sledi izdelava sporočila XML, proces serializacije sporočila XML in dodajanja ovojnice SOAP. Nato podpišemo celoten dokument skupaj z ovojnico in pošljemo zahtevo. Če sprejmemo odgovor v določenem času, poskusimo prebrati podatek EOR in ga posredovati kot odgovor. Sicer sporočimo neuspešno potrjevanje.




Slika 30: Diagram aktivnosti davčnega potrjevanja.

### 3.6.3 Diagram aktivnosti izdelave naročila

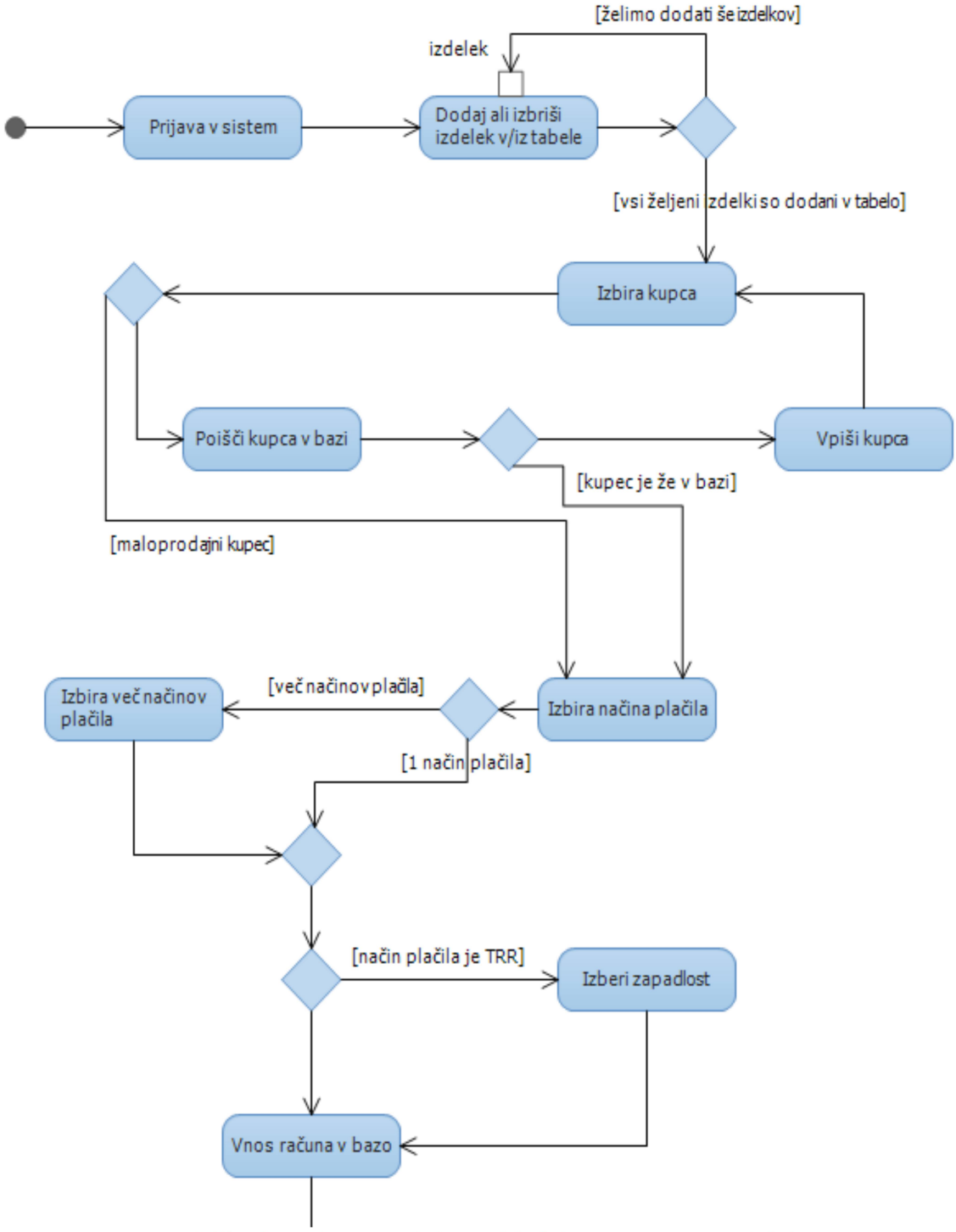
Diagram aktivnosti na spodnji sliki 31 prikazuje zaporedje korakov pri izdelavi naročila pri prihodu kupca oz. pri dodatnem naročanju izdelkov. Sam proces je že podrobno opisan v poglavju 3.3.1 Diagram zaporedja izdelave naročila.



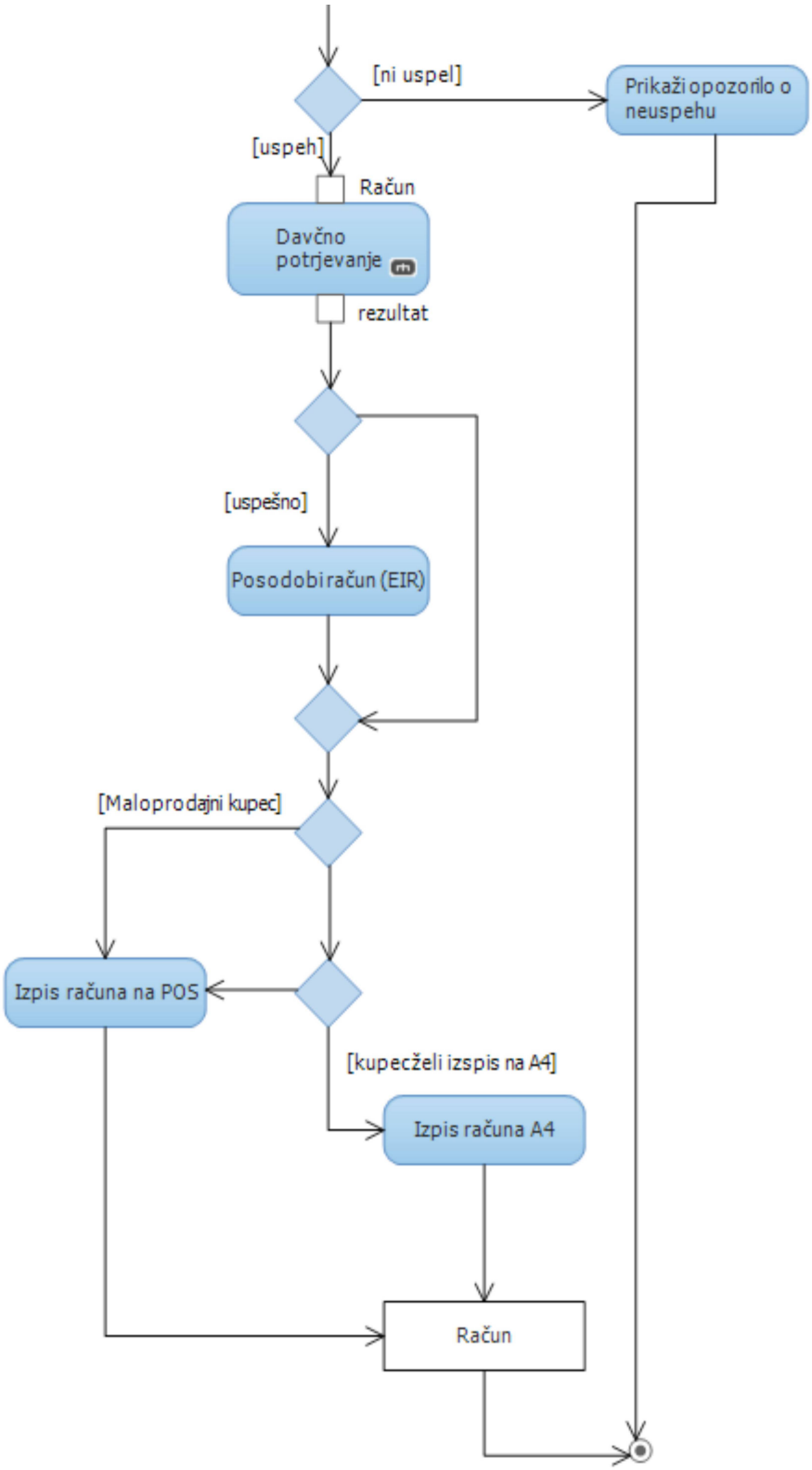

Slika 31: Diagram aktivnosti izdelave naročila.

### 3.6.4 Diagram aktivnosti izdelave računa

Opis aktivnosti, prikazanih na slikah 32 in 33, je podan v poglavju 3.3.2 Diagram zaporedja izdelave računa.



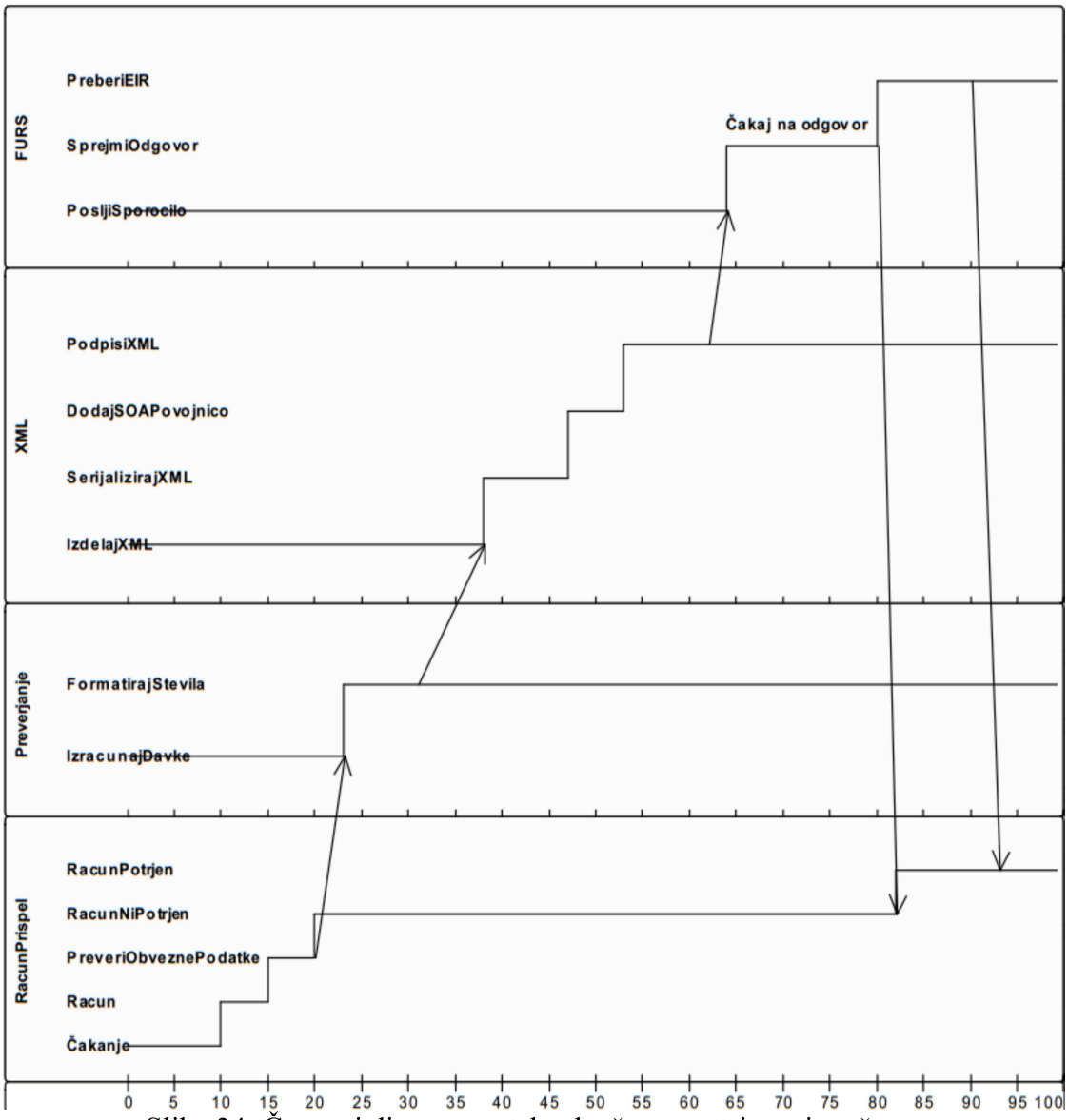

Slika 32: Diagram aktivnosti izdelave računa – prvi del.




Slika 33: Diagram aktivnosti izdelave računa – drugi del.

## 3.7 Časovni diagram poteka davčnega potrjevanja

Slika 34 opisuje časovni potek davčnega potrjevanja računa. Postopek davčnega potrjevanja tako lahko proučujemo z dveh različnih vidikov. Prvega smo že podali z diagramom aktivnosti v poglavju 3.6.2 Diagram aktivnosti davčnega potrjevanja. Čas, ki je potreben za obdelavo celotne aktivnosti, seveda ni fiksen. Odvisen je od števila izdelkov, količine izdelkov in pripadnosti različnim davčnim skupinam, vendar pa vseeno lahko rečemo, da je do faze PosljiSporocilo za uporabnika, ki čaka na potrditev, vedno enak. Od tukaj naprej, ko pošljemo sporočilo, moramo čakati na odgovor DURS-a, pri čemer pa ne moremo vplivati na hitrost. Tukaj lahko pride do izpada povezave ali do predolge obdelave, kadar moramo prekiniti čakanje zaradi dejstva, da kupec čaka na račun. Sam odgovor po navadi pridobimo v 1—2 sekundah. Aplikacija ima možnost parametrizacije čakalnega časa. Privzeto je čakalni čas nastavljen na 12 sekund.




Slika 34: Časovni diagram poteka davčnega potrjevanja računa.

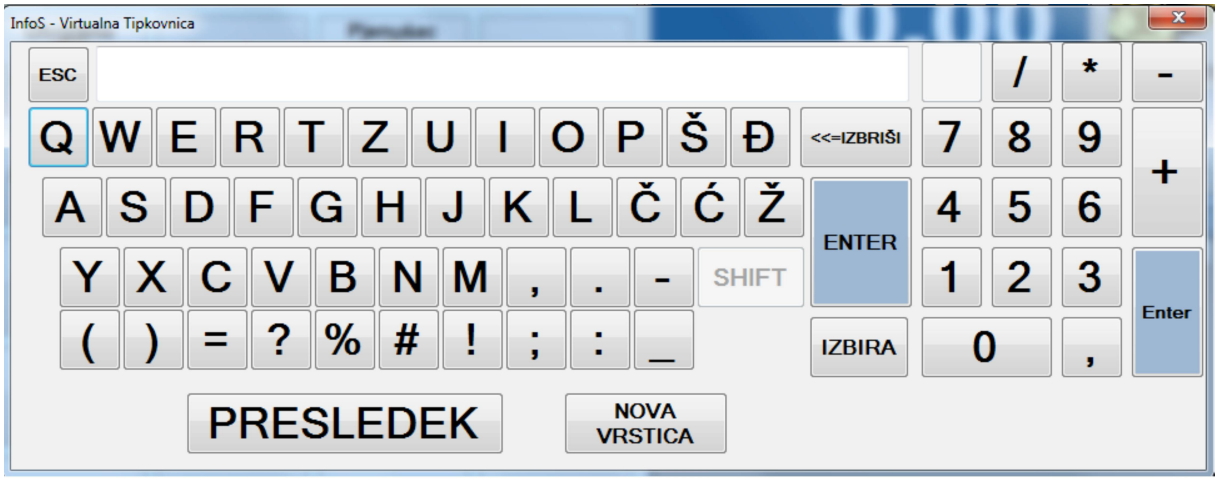
# 4 Uporaba aplikacije (zaslonska okna)

Opis zaslonskih oken je podan samo tam, kjer že sama slika samoumevno ne pove, kakšna je funkcionalnost opazovanega okna.

Slika 35 prikazuje zaslonsko okno za prijavo v aplikacijo. Prijavno okno se prek parametrov aplikacije lahko spremeni tako, da se vpisuje tudi samo PIN (namesto Uporabniškega imena in Gesla). To je po navadi takrat, ko se aplikacija uporablja na napravah z zaslonom, občutljivim na dotik, kjer je treba čim bolj poenostaviti proces vnosa podatkov.


Slika 35: Prijava v aplikacijo.

Če se aplikacija uporablja le na teh napravah, lahko tudi s parametrom nastavimo, da se s klikom oz. dotikom polja za vnos besedila odpre virtualna tipkovnica, kot je prikazano na sliki 36. Analogija velja za vsa tekstualna polja za vnos.



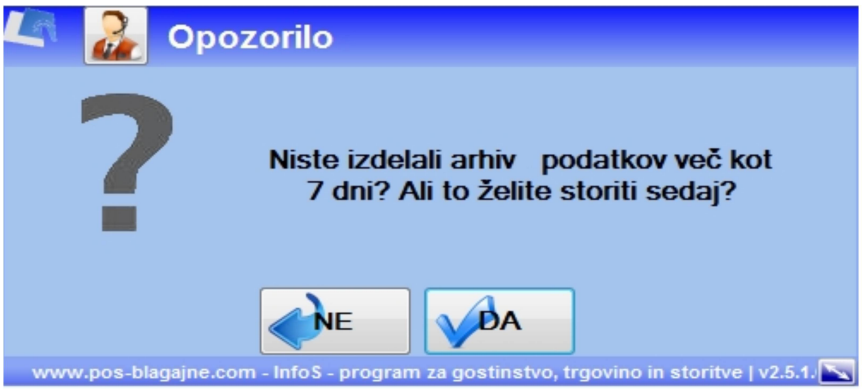

Slika 36: Virtualna tipkovnica.

Preden se uporabniku pokaže glavno okno blagajne, mora še eksplicitno potrditi trenutni datum. Ta se avtomatsko ponudi iz sistemskega datuma računalnika. Vseeno pa mora uporabnik imeti možnost spremembe (slika 37), če pride do izpada sistemske ure ali če nekdo pomotoma na računalniku zamenja datum.

Slika 37: Izbira datuma.

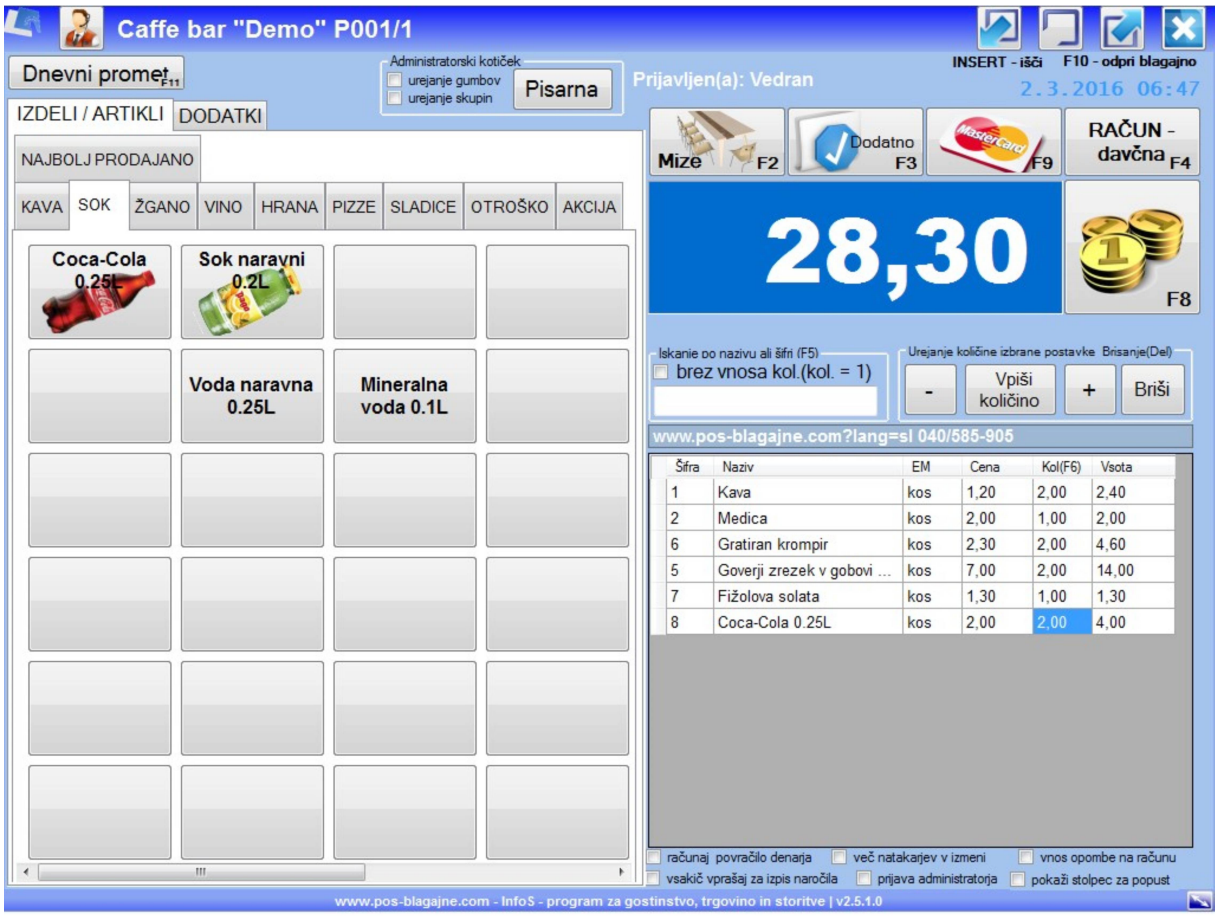
Iz izkušenj, pridobljenih pri različnih uporabnikih, lahko ugotovimo, da so uporabniki nepazljivi pri branju opozoril in preverjanju določenih podatkov, ki so zelo pomembni za pravilno delovanje in skladnost podatkov. Primer na sliki 39 je rezultat takšnega obnašanja, ker je veliko uporabnikov zgoraj opisano možnost preverjanja in izbire datuma enostavno spregledalo in so izdajali račune z datumom v preteklosti. Po navadi so krivdo prestavili prav na razvijalce aplikacije. Zato je dodano še naslednje preverjanje, ki še enkrat eksplicitno zahteva preverjanje datuma in časa. Slednji je postal še pomembnejši pri davčnem potrjevanju računov, ker se računi pošiljajo v realnem času, in mora biti podatek o času (in datumu) čim bolj natančen.

Slika 39: Preverjanje datuma in časa.




Slika 40: Opozorilo za arhiv podatkovne baze.

Slika 41 prikazuje glavno okno blagajne. To je zaslon, s katerem se bo akter – natakar največkrat srečal. Na njem lahko izdeluje naročila, ponudbe, račune, storno račune in vpisuje kupce. Postopek izdelave določenega tipa dokumenta se začne z dodajanjem izdelkov v tabelo na desni strani. To poteka tako, da uporabnik klikne na določeno skupino in določen izdelek. V tabeli lahko direktno spreminja količine, popuste ali izbriše izdelek, preden zaključi dokument. Artikle lahko dodamo tudi po šifri ali nazivu z vpisom podatkov v sredinsko polje, če je tako uporabniku lažje. Na koncu se v odvisnosti od tipa dokument zaključi z dodajanjem na mizo ali z izbiro načina plačila ali kupca.




Slika 41: Glavno okno blagajne.


Slika 42: Okno izbire načina plačila.

Demo potrjevanje d.o.o.
Dunajska cesta 105, 1000 Ljubljana
DŠ: 39253074

**Caffe bar "Demo"**

Ulica 7, Maribor

RAČUN P001-1-26
Način plačila: GOTOVINA
Datum 2.3.2016 Čas: 06:51:58
Streže Vas:Vedran

| Artikel | Cena | Kol | Skupaj (EUR) |
|---|---|---|---|
| Kava | 1,20 | 2,00 | 2,40 |
| Medica | 2,00 | 1,00 | 2,00 |
| Gratiran krompir | 2,30 | 2,00 | 4,60 |
| Goverji zrezek v gob | 7,00 | 2,00 | 14,00 |
| Fižolova solata | 1,30 | 1,00 | 1,30 |
| Coca-Cola 0.25L | 2,00 | 2,00 | 4,00 |

**ZA PLAČILO: 28,30 EUR**

DAVKI:

| Naziv | Osnova | Stopnja (%) | Znesek (EUR) |
|---|---|---|---|
| DDV | 24,02 | 9,50 | 2,28 |
| DDV | 1,64 | 22,00 | 0,36 |

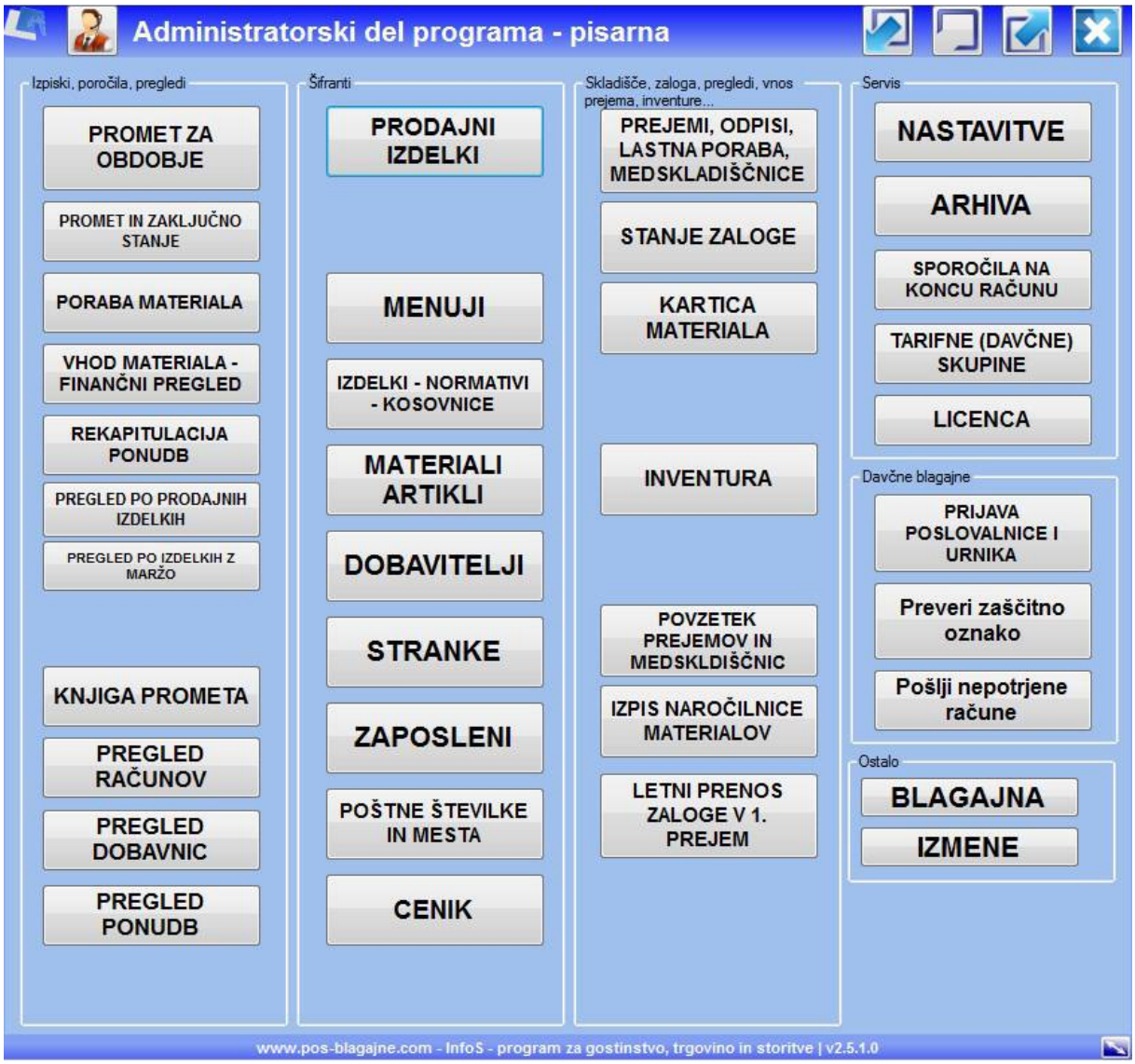
www.pos-blagajne.com?lang=sl 040/585-905

EOR: 048b3a75-785b-4fcd-bbd7-3f924e3c6dcd
ZOI: 0834ddbc6bcd1b830d65e8192af016d9

Slika 43: Primer izpisanega davčno potrjenega računa na tiskalniku POS.

Pisarniški del programa je namenjen voditeljem, ki se bodo največkrat srečali ravno s tem oknom. Tukaj lahko voditelj ureja šifrante, vhodno-izhodne dokumente in preverja zalogo. Ima dostop tudi do že izdanih računov, ponudb, dobavnic in raznih pregledov prodaje. Vsak pregled, urejanje ali izdelava določenega dokumenta poteka prek lastnih zaslonskih oken. Ta zaradi velika števila niso opisana v nadaljevanju. Podroben opis tudi ni potreben za razumevanje delovanja aplikacije, ker lahko uporabnik zelo hitro pride z uporabo, tj. kliki po želenih gumbih, do potrebne informacije.

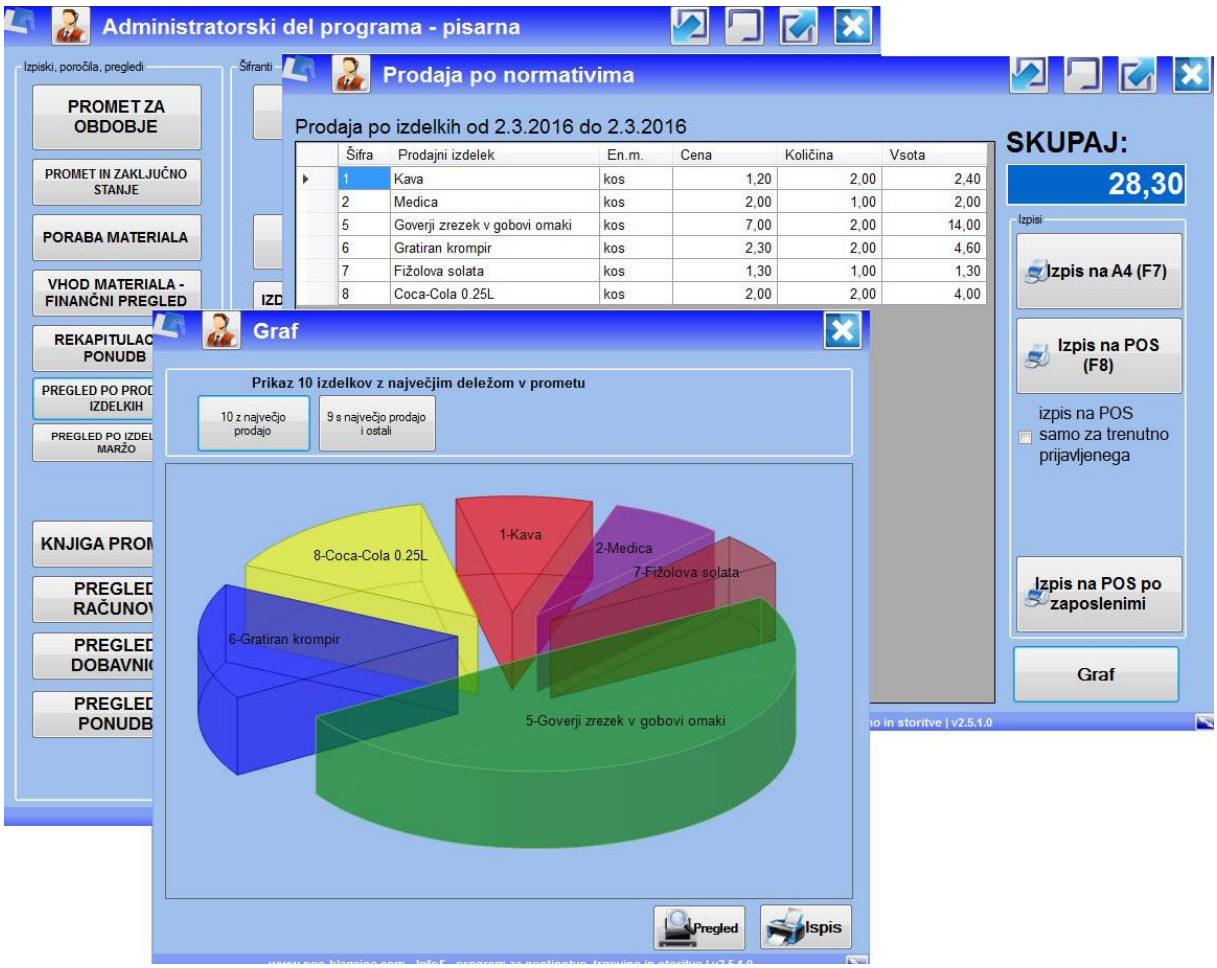
Večina funkcionalnosti gumbov je že opisana v že prej opisanem poglavju 3.1 Primeri uporabe – zmožnosti aplikacije.




Slika 44: Pisarna – *back office* – del programa za voditelje.

Kot primer je na sliki 45 podan primer pregleda prometa za izbrano obdobje po prodanih izdelkih v gostinstvu. V tabeli je prikazan promet, razvrščen po posameznem prodajnem izdelku. Tabelo lahko razvrščamo s klikom na katerokoli glavo stolpca. Uporabnik lahko prikazano tabelo izpiše na navaden tiskalnik A4, če ga pa nima nameščenega v lokalu, lahko izpiše promet na tiskalnik POS. Izpis se lahko naredi za vse zaposlene oz. za vse uporabnike, ki so prodajali izdelke, ali pa samo za trenutno prijavljene v aplikacijo. Slednje je koristno pri preverjanju stanja zaloge pri izmenah.

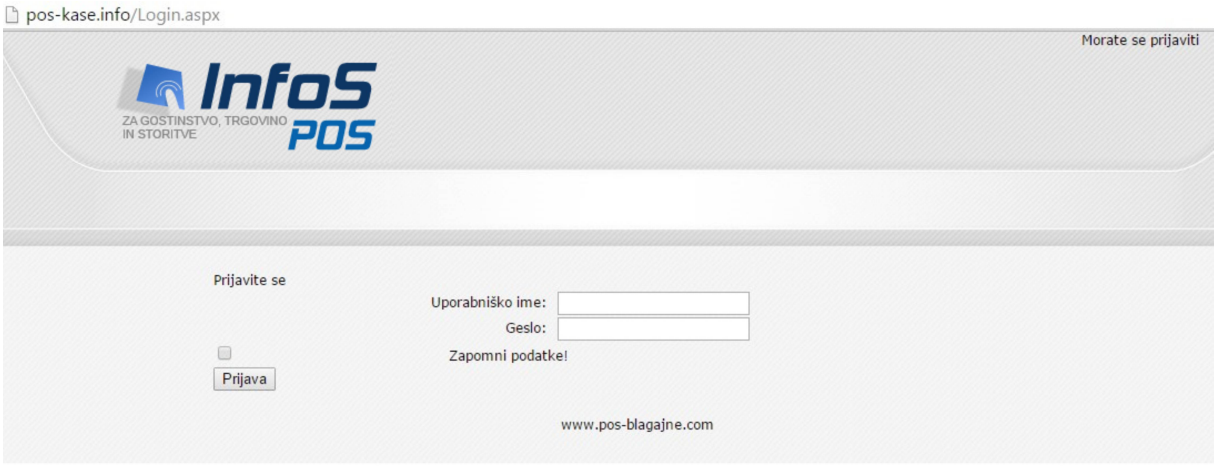
Vsebina tabele se lahko tudi grafično ponazori s pritiskom na gumb Graf, ki prikaže deset najbolj prodajanih izdelkov v opazovanem obdobju. Graf nudi tudi možnost pregleda devetih najbolj prodajanih artiklov in vseh ostalih artiklov, ki so vsebovani v eni rezini grafa. Graf lahko natisnemo na navaden tiskalnik.



Slika 45: Primer pregleda prodaje izdelkov.

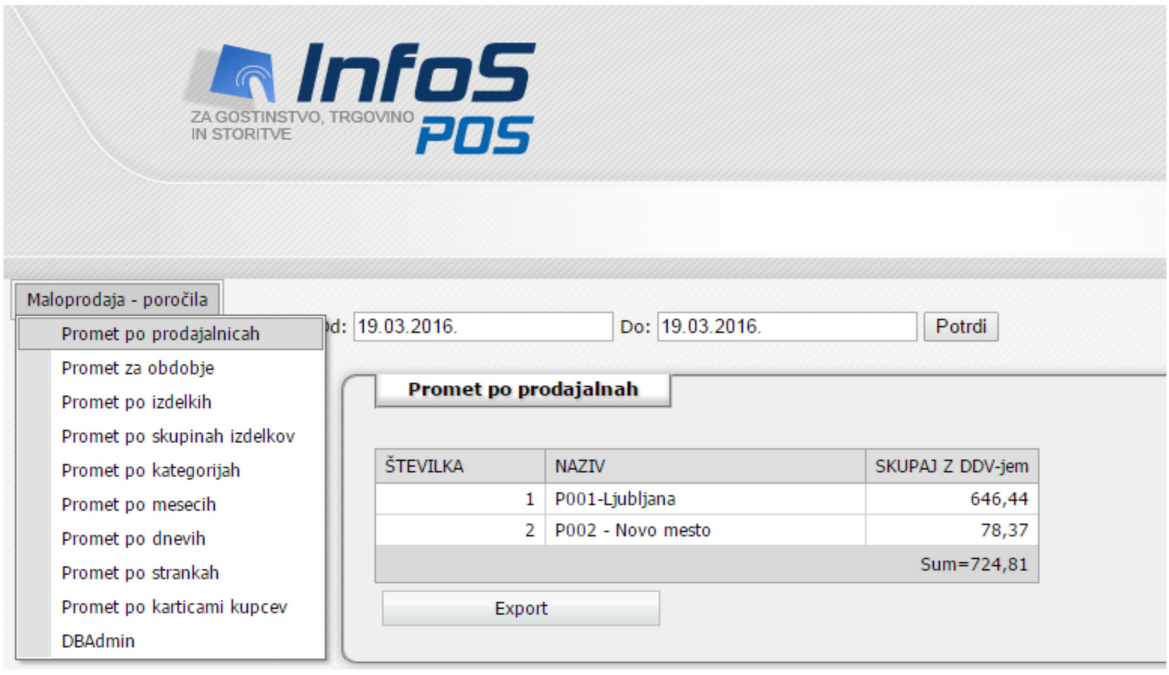
# 5 Spletna aplikacija

Spletna aplikacija omogoča dostop in pregled podatkov ter poročil iz katerekoli lokacije. Pogoj je le dostop do interneta. Z vnosom uporabniškega imena in gesla, kot je prikazano na sliki 46, bo sistem preveril, če takšen uporabnik obstaja. Če uporabnik obstaja, bo sistem iz podatkovne baze prebral davčno številko, za katero ima uporabnik omogočen pregled, in se mu na podlagi dodeljenih pravic prikažejo možnosti ter navigacija na levi strani.




Slika 46: Prijava v spletno aplikacijo.

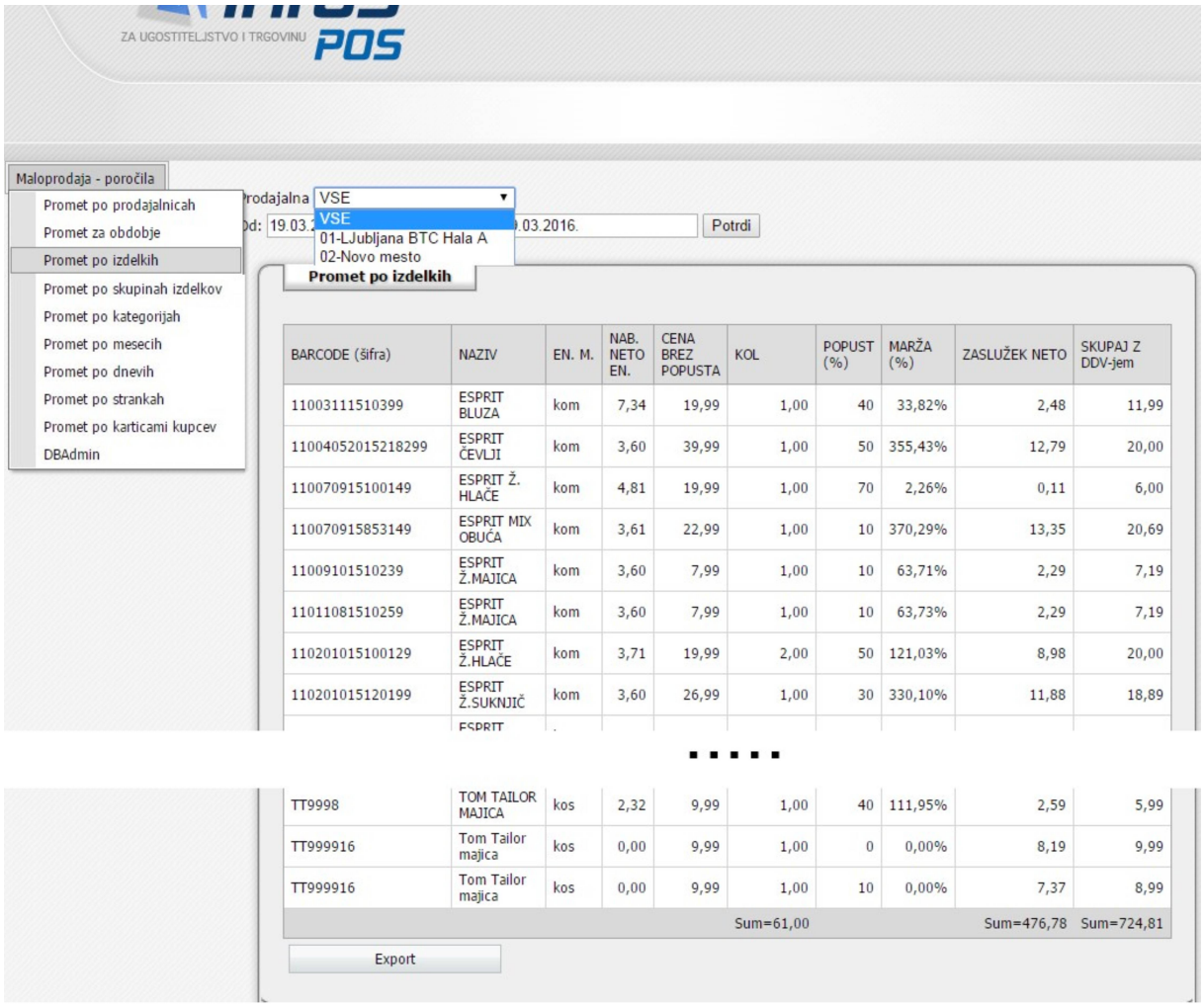
Spletna aplikacija nudi različne preglede in poročila ter izvoz podatkov v poljubnem formatu. Na sliki 47 je prikazan pregled prometa za izbrano obdobje po prodajalnah. Na levi strani so prikazana tudi ostala možna poročila za trenutno pripravljenega uporabnika glede na njegove pravice.




Slika 47: Pregled prodaje po prodajalnah.

Eden najbolj uporabnih je tudi pregled po izdelkih, kjer se zraven vsakega izdelka poleg količine in prodajne cene izpišeta tudi realna neto marža in morebitni popust. Poleg tega je jasno podan tudi podatek o neto zaslužku iz prodaje (brez vračunanih odvisnih in ostalih stroškov prodaje, osebja in podobno). To omogoča odločilnim v poslovnem sistemu izločiti izdelke z največjo dodano vrednostjo in povečati njihovo prisotnost v prodajalnah, enako pa tudi tiste, ki delajo izgubo poslovnemu sistemu. Slika 48 prikazuje takšen pregled prodaje po izdelkih z maržo.

Na vseh pregledih lahko razvrščamo po vseh stolpic naraščajoče ali padajoče, tako da enostavno kliknemo na glavo stolpca. Stolpce lahko tudi strnemo ali razširimo in jim spreminjamo vrstni red z uporabo funkcionalnosti primi in spusti (*drag & drop*).

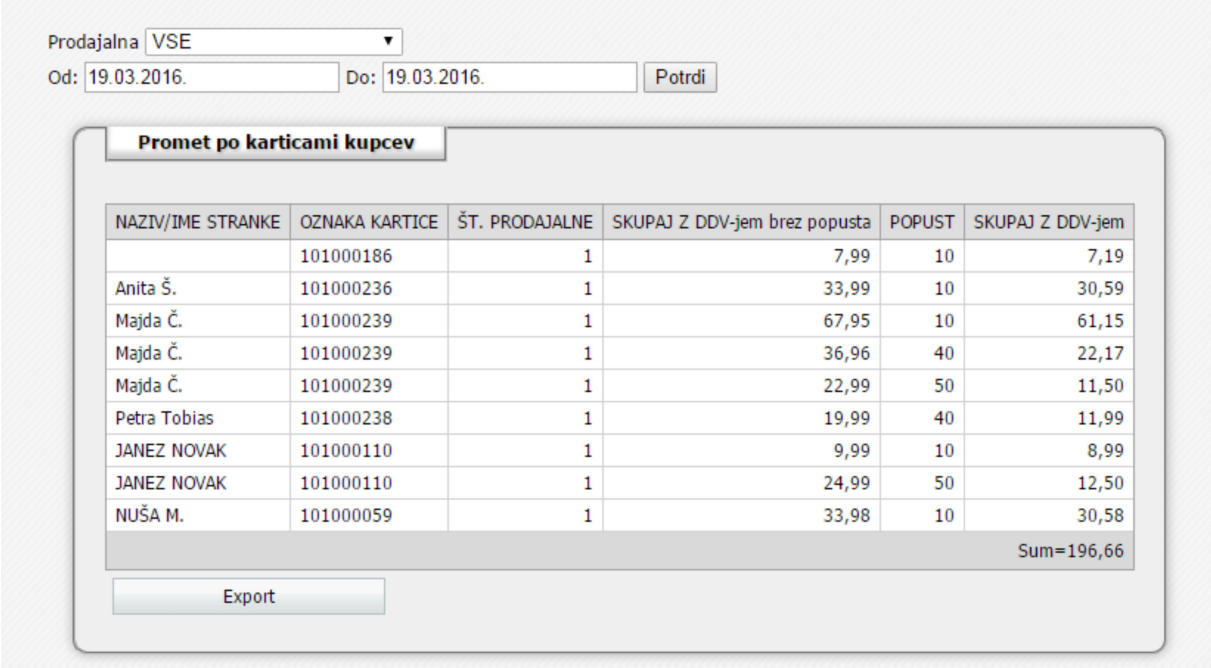



| BARCODE (šifra) | NAZIV | EN. M. | NAB. NETO EN. | CENA BREZ POPUSTA | KOL | POPUST (%) | MARŽA (%) | ZASLUŽEK NETO | SKUPAJ Z DDV-jem |
|---|---|---|---|---|---|---|---|---|---|
| 11003111510399 | ESPRIT BLUZA | kom | 7,34 | 19,99 | 1,00 | 40 | 33,82% | 2,48 | 11,99 |
| 11004052015218299 | ESPRIT ČEVLJI | kom | 3,60 | 39,99 | 1,00 | 50 | 355,43% | 12,79 | 20,00 |
| 110070915100149 | ESPRIT Ž. HLAČE | kom | 4,81 | 19,99 | 1,00 | 70 | 2,26% | 0,11 | 6,00 |
| 110070915853149 | ESPRIT MIX OBUĆA | kom | 3,61 | 22,99 | 1,00 | 10 | 370,29% | 13,35 | 20,69 |
| 11009101510239 | ESPRIT Ž.MAJICA | kom | 3,60 | 7,99 | 1,00 | 10 | 63,71% | 2,29 | 7,19 |
| 11011081510259 | ESPRIT Ž.MAJICA | kom | 3,60 | 7,99 | 1,00 | 10 | 63,73% | 2,29 | 7,19 |
| 110201015100129 | ESPRIT Ž.HLAČE | kom | 3,71 | 19,99 | 2,00 | 50 | 121,03% | 8,98 | 20,00 |
| 110201015120199 | ESPRIT Ž.SUKNJIČ | kom | 3,60 | 26,99 | 1,00 | 30 | 330,10% | 11,88 | 18,89 |
| TT9998 | TOM TAILOR MAJICA | kos | 2,32 | 9,99 | 1,00 | 40 | 111,95% | 2,59 | 5,99 |
| TT999916 | Tom Tailor majica | kos | 0,00 | 9,99 | 1,00 | 0 | 0,00% | 8,19 | 9,99 |
| TT999916 | Tom Tailor majica | kos | 0,00 | 9,99 | 1,00 | 10 | 0,00% | 7,37 | 8,99 |
| | | | | | Sum=61,00 | | | Sum=476,78 | Sum=724,81 |

Slika 48: Pregled prodaje po izdelkih z maržo in zaslužkom.

Za vse preglede lahko izberemo pregled za celoten poslovni sistem (za vse prodajalne ali profitne centre) ali pa pregled za izbrano prodajalno ali profitni center. Zaradi prevelike dolžine je srednji del slike izrezan, dodan pa je samo zaključek oz. vsota vrstic.

Za vsak poslovni objekt so zelo pomembni tudi stalni kupci, ker so oni tisti, ki največkrat v obdobju slabe prodaje omogočajo preživetje poslovnemu sistemu. Zato ima tudi aplikacija lokalne blagajne vgrajen sistem nagrajevanja, ki je odvisen od posameznega poslovnega sistema. Največkrat je to v obliki popustov ali darilnih izdelkov. V sistemih, kjer stalnim kupcem podelijo kartico ali številko kupca za uveljavitev dodatnih ugodnosti, bo aplikacija lokalno (pa tudi prek sinhronizacije v oblak) beležila nakup vsakega takšnega kupca. Zato je tudi na spletu omogočen pregled prodaje po karticah kupcev. To omogoča odločilnim v procesu nagrajevanja lažje odločitve glede dodatnih ugodnosti za posameznega kupca ali skupino kupcev. Poleg osnovnih informacij o kupcu se pri izdelavi kartice in sprejemu dovoljenja za tržno komuniciranje od kupca pridobi tudi naslov ali naslov elektronske pošte. Na takšen način je potem lažja komunikacija s kupci in predstavitev novih izdelkov, akcij, posebnih ponudb in podobno. Slika 49 prikazuje takšen pregled. S slike je razvidno tudi, da naziv/ime prve stranke manjka, kar pomeni, da so računi sinhronizirani, podatek o kupcu pa še ni sinhroniziran iz lokalne aplikacije (gre za novo dodanega kupca, ki ga komponenta AOP še ni sinhronizirala na spletni strežnik).

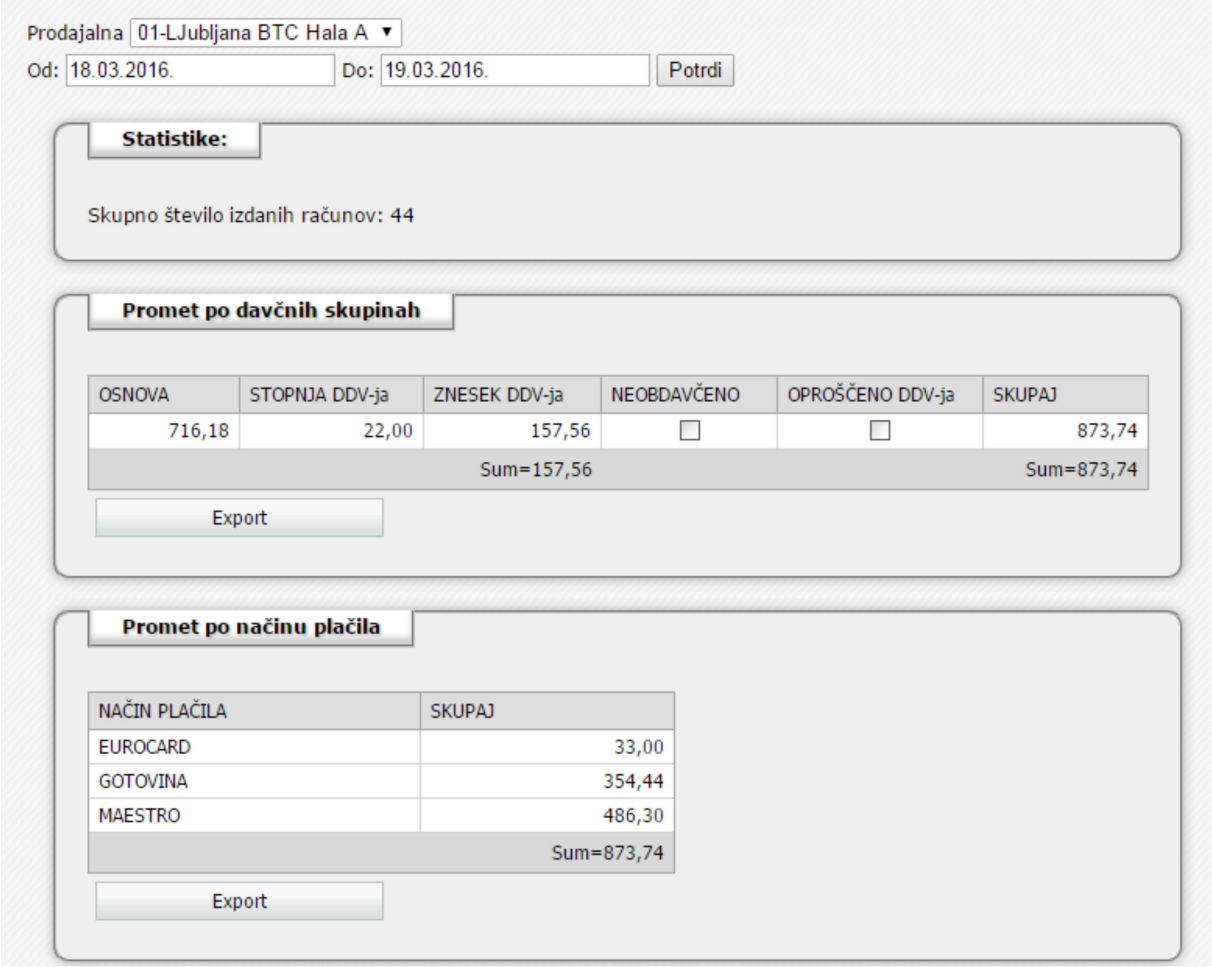



| NAZIV/IME STRANKE | OZNAKA KARTICE | ŠT. PRODAJALNE | SKUPAJ Z DDV-jem brez popusta | POPUST | SKUPAJ Z DDV-jem |
|---|---|---|---|---|---|
| | 101000186 | 1 | 7,99 | 10 | 7,19 |
| Anita Š. | 101000236 | 1 | 33,99 | 10 | 30,59 |
| Majda Č. | 101000239 | 1 | 67,95 | 10 | 61,15 |
| Majda Č. | 101000239 | 1 | 36,96 | 40 | 22,17 |
| Majda Č. | 101000239 | 1 | 22,99 | 50 | 11,50 |
| Petra Tobias | 101000238 | 1 | 19,99 | 40 | 11,99 |
| JANEZ NOVAK | 101000110 | 1 | 9,99 | 10 | 8,99 |
| JANEZ NOVAK | 101000110 | 1 | 24,99 | 50 | 12,50 |
| NUŠA M. | 101000059 | 1 | 33,98 | 10 | 30,58 |
| | | | | | Sum=196,66 |

Slika 49: Prodaja po karticah kupcev.

Kot prihodke za knjigovodstvo in finančni oddelek poslovnega sistema bo zagotovo največkrat zadoščal izpis prometa za določeno obdobje, agregirano po davčnih skupinah in načinu plačila. Slika 50 prikazuje takšen pregled prometa. Tukaj so podatki o prodaji razdeljeni po načinu plačila, ki pomaga knjigovodsko uskladitev različnih kont. Drugi izpis pa prikazuje prodajo, razvrščeno po davčnih skupinah. Davčna skupina ima določeno stopnjo DDV-ja, podatek, ali gre za neobdavčene izdelke/storitve oz. ali so posamezne skupine oproščene DDV-ja po določenem členu zakona. Vsak podatek je sestavljen iz osnove, zneska

davka in skupnega zneska z vključenim davkom. Seštevek zneska z vključenim davkom mora biti enak seštevku po načinih plačila.



Slika 50: Pregled prodaje po davčnih skupinah in načinu plačila.

# 6 Sklepne ugotovitve

Z izdelanim modelom aplikacije, ki vsebuje različne diagrame UML statičnega in dinamičnega pogleda na sistem ter opise posameznih diagramov in gradnikov, je bil iz končnega izdelka pridobljen pregled aplikacije z nekoliko višjega in bolj abstraktnega nivoja, kot je bilo do sedaj. S pridobljenim modelom se bo o možnostih, modulih in tehničnih rešitvah lahko v razmeroma kratkem času seznanil katerikoli novi član razvojne skupine. To bo posredno razbremenilo vodilne ljudi na projektu in jim omogočilo osredotočanje na reševanje drugih problemov, za katere so zadolženi.

Izdelani diagrami aktivnosti in primeri uporabe bodo zagotovo pomagali partnerjem pri lažjem razumevanju tako možnosti celotne aplikacije kot tudi pri poteku določenih procesov v ozadju delovanja aplikacije. To bo neposredno vplivalo na boljšo tržno komunikacijo in posredno na boljšo prodajo v prihajajočih časih. Iz komponentnega diagrama lahko razberemo, da bi bilo treba v prihodnosti razmišljati o ločitvi določenih, sedaj skupaj sestavljenih delov sistema. To bo omogočilo lažje vzdrževanje in hitrejšo posodobitev novejših različic aplikacije, kar bi pozitivno vplivalo na zadovoljstvo končnih uporabnikov.

Iz opisanega lahko sklepamo, da čeprav se na področju razvoja programske opreme trenutno nahajamo v obdobju agilnosti, kjer je poudarek na uporabnikih in hitrem razvoju, lahko nedvoumno rečemo, da je treba zraven kode izdelati vsaj štiri dodatne dokumente za boljše razumevanje sistema, in sicer:

1. arhitekturo sistema (da lahko člani dobijo grobo sliko sistema),
2. model problemske domene (da lahko člani razumejo koncept problemske domene),
3. ključne primere uporabe (da lahko člani razumejo uporabljene koncepte),
4. postavitev sistema (da lahko člani takoj vidijo možne načine postavitve komponent).

*“While code is the truth, it is not the whole truth.”*

- Grady Booch – eden od avtorjev UML-ja

# Literatura

[1] Arie van Deursen et al., Symphony: View-Driven Software Architecture Reconstruction, 2005

[2] Jean-Marie Favre, CacOphoNy: Metamodel-Driven Software Architecture Reconstruction, University of Grenoble, France, 2004

[3] Rick Kazman, Liam O'Brien, Chris Verhoef, Architecture Reconstruction Guidelines, Third Edition, Software Engineering Institute - Carnegie Mellon University, 2003

[4] René L. Krikhaar, Software Architecture Reconstruction, Philips Research Laboratories, 1999

[5] Liam O'Brien, Christoph Stoermer, Chris Verhoef, Software Architecture Reconstruction: Practice Needs and Current Approaches, Software Engineering Institute - Carnegie Mellon University, 2002

[6] Claudio Riva et al., UML-based Reverse Engineering and Model Analysis Approaches for SoftwareArchitecture Maintenance , Nokia Research CenterSoftware Technology Laboratory, Tampere University of TechnologyInstitute of Software Systems, Finland, 2004

[7] Microsoft Visual Studio. Dostopno na: https://www.visualstudio.com/

[8] UML diagrams. Dostopno na: http://www.uml-diagrams.org/

[9] Unified modeling language. Dostopno na: http://www.uml.org

[10] XP - eXtreme Programming. Dostopno na: http://wiki.expertiza.ncsu.edu/index.php/CSC/ECE_517_Fall_2012/ch2a_2w8_vp

[11] .NET Framework . Dostopno na: https://msdn.microsoft.com/en-us/library/z1zx9t92.aspx

[12] .NET Framework Conceptual Overview. Dostopno na: https://msdn.microsoft.com/en-us/library/zw4w595w(v=vs.100).aspx